Automated Support for Performance and Energy Evaluation for Cloud Applications

by

Feifei Chen

A thesis submitted to
Swinburne University of Technology

for the degree of
Doctor of Philosophy

September, 2015
To my parents and my husband
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Feifei Chen

September, 2015
Acknowledgements

PhD is a once-in-a-lifetime opportunity and experience. It is tough at times and may feel like an eternity, but it teaches you a lot, and I am truly happy that I have had a chance to complete it. It would not have happened without all those people who helped me along the way.

I sincerely express my deepest gratitude to my coordinating supervisor Professor John Grundy, my associate supervisors, Professor Yun Yang and Associate Professor Jean-Guy Schneider, for their support, supervision and continuous encouragement throughout my PhD study. Without their consistent support, I would not have been able to complete this research work. Honestly, I have always counted myself very lucky to have all of them as my supervisors.

I thank the Australian Research Council for supporting me with a full Research Scholarship throughout my doctoral program. I also thank the Research Committee of the School of Software and Electrical Engineering for research publication funding support and for providing me with financial support to attend conferences.

I am deeply grateful to my parents Zhiguo Chen and Tingting Yao for raising me up, teaching me to be a good person, and supporting me to study abroad. I also thank my sisters Juan Chen and Min Chen, for their great support. Last but not least, I thank my husband Qiang He, for his love, understanding, encouragement, sacrifice and help. No words can express my love and sincere gratitude towards him. I could not achieve anything in my life and this thesis is not an exception without him.
Abstract

Cloud computing is a new and promising computing paradigm which delivers computing infrastructure as a utility. The cloud model delivers highly scalable, shared computing resources which supports conventional in-house applications in migrating to the cloud. However, the proliferation of cloud computing has resulted in the establishment of large-scale data centres. Such data centres consume huge amounts of energy. In the meantime, they must fulfil the need to meet ever-increasing system performance and other Quality of Service (QoS) requirements for cloud services. Thus, cloud application deployment and management solutions that guarantee system performance and SLAs with minimum energy consumption need to be developed for cloud service providers.

This thesis provides a set of solutions that can be applied to determine energy-efficient cloud application deployment strategies based on collected performance and energy consumption data. First, a model is developed to profile and analyse the energy consumption of cloud applications. This new energy consumption model provides the basis for characterising the energy consumption of cloud applications under different system configurations and application workloads. Second, in order to validate the energy consumption model, extensive experiments were conducted to profile and analyse energy consumption and system performance of several real-world cloud applications under different application workloads and system configurations. Third, a series of energy consumption patterns have been identified from these experimental results, which are formalised as “signatures” using the Object Constraint Language (OCL). These energy consumption patterns can be used to predict the energy efficiency of cloud applications without running expensive and time-consuming load tests. Finally, an automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments has been developed, named StressCloud. It integrates the energy consumption model and signatures to support 1) automatic, empirical profiling of
system performance and energy consumption of cloud applications; and 2) pre-test analysis of the trade-off between cloud application energy consumption and performance.

Three case studies have been conducted on significant cloud applications to validate our approach. Experimental results show that our approach can efficiently and correctly analyse the trade-off between system performance and energy consumption of different cloud applications in real-world environments.
The Author’s Publications

Published:


# Table of Content

Declaration.............................................................................................................................................. i

Acknowledgements........................................................................................................................... ii

Abstract............................................................................................................................................... iii

The Author’s Publications .............................................................................................................. v

Table of Content List of Figures .......................................................................................... vi

List of Figures .................................................................................................................................. x

List of Tables .................................................................................................................................... xv

Chapter 1  Introduction .................................................................................................................. 1

1.1 Research Background and Motivation .................................................................................. 1

1.2 Key Research Issues .............................................................................................................. 4

1.3 Key Contributions .................................................................................................................. 6

1.4 Overview of Thesis ............................................................................................................... 8

Chapter 2  Literature Review ..................................................................................................... 10

2.1 Introduction ........................................................................................................................... 10

2.2 Cloud Computing ................................................................................................................. 11

2.3 Cloud System Performance Evaluation .............................................................................. 13

2.3.1 Performance Monitoring ............................................................................................... 14

2.3.2 Simulation-based Modelling ........................................................................................... 15

2.3.3 Test-beds .......................................................................................................................... 18

2.3.4 Discussion ......................................................................................................................... 20

2.4 Energy Efficiency in Cloud Systems .................................................................................... 22

2.4.1 Power and Energy Consumption Models ......................................................................... 22

2.4.2 Energy Saving Policies ...................................................................................................... 25

2.4.3 Energy Profiling and Analysis ......................................................................................... 28

2.5 Summary .................................................................................................................................. 30

Chapter 3  Approach ....................................................................................................................... 32

3.1 Overview .................................................................................................................................. 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Analysing Performance and Energy Consumption of Cloud Applications</td>
<td>34</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Energy Consumption Model</td>
<td>34</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Performance and Energy Consumption Trade-Off Analysis</td>
<td>35</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Energy Consumption Signature</td>
<td>37</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Performance and Energy Consumption Analysis Tool for Cloud Applications</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>Summary</td>
<td>41</td>
</tr>
</tbody>
</table>

**Chapter 4**  
Energy Consumption Model | 42

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Modelling Energy Consumption of Cloud Applications</td>
<td>43</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Model Setting</td>
<td>43</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Energy Calculation Unit</td>
<td>44</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Energy Calculation Formula</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>Example Usage</td>
<td>47</td>
</tr>
<tr>
<td>4.4</td>
<td>Discussion</td>
<td>51</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary</td>
<td>53</td>
</tr>
</tbody>
</table>

**Chapter 5**  
Performance and Energy Consumption Profiling and Analysis | 54

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>Experimental Setup</td>
<td>55</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Test-bed</td>
<td>55</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Profiling Method</td>
<td>58</td>
</tr>
<tr>
<td>5.3</td>
<td>Test Case Design</td>
<td>59</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Computation-Intensive Task</td>
<td>59</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Data-Intensive Task</td>
<td>61</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Communication-Intensive Task</td>
<td>64</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Mixed Computation- and Data-Intensive Task</td>
<td>64</td>
</tr>
<tr>
<td>5.3.5</td>
<td>Mixed Computation-, Data- and Communication-Intensive Task</td>
<td>65</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiment Results Analysis</td>
<td>66</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Computation-Intensive Workloads</td>
<td>66</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Data-Intensive Workloads</td>
<td>76</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Communication-Intensive Workloads</td>
<td>86</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Mixed Computation- and Data-Intensive Workloads</td>
<td>89</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Mixed Computation-, Data- and Communication-Intensive Workloads</td>
<td>92</td>
</tr>
<tr>
<td>5.5</td>
<td>Discussion and Conclusion</td>
<td>95</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>98</td>
</tr>
</tbody>
</table>
Chapter 6  Energy Efficiency Analysis of Cloud Applications using Formalised Signatures ................................................................. 99
6.1 Introduction ............................................................................................................. 99
6.2 Energy Efficiency Pattern Definition Schema .................................................... 102
  6.2.1 Cloud Application Workload Model ............................................................. 104
  6.2.2 Cloud System Architecture Model ............................................................... 108
6.3 Energy Efficiency Patterns and Signature Specification ..................................... 109
6.4 Discussion .......................................................................................................... 115
6.5 Summary ............................................................................................................. 116

Chapter 7  StressCloud – Automating Performance and Energy Consumption Analysis .................................................................................. 117
7.1 Introduction .......................................................................................................... 117
7.2 Overview of StressCloud ..................................................................................... 119
7.3 StressCloud Example Usage – JPetStore ............................................................ 123
  7.3.1 Cloud Application Workload Modelling ....................................................... 123
  7.3.2 Cloud Architecture Modelling ...................................................................... 127
  7.3.3 Model Validation .......................................................................................... 128
  7.3.4 Energy Efficiency Analysis Using Formalised Signatures ........................... 130
  7.3.5 Deployment Plan Generation ........................................................................ 133
  7.3.6 Load Test Plan Generation .......................................................................... 134
  7.3.7 Load Tests Running and Results Visualising ................................................ 136
7.4 Design and Implementation .................................................................................. 137
7.5 Evaluation ............................................................................................................. 140
  7.5.1 Experimental Validation ............................................................................... 140
  7.5.2 User Evaluation ............................................................................................. 141
  7.5.3 Threats to Validity .......................................................................................... 144
7.6 Discussion .......................................................................................................... 145
7.7 Summary ............................................................................................................. 148

Chapter 8  Case Studies ........................................................................................ 149
8.1 Introduction .......................................................................................................... 150
8.2 Experimental Setup .............................................................................................. 151
8.3 Case Study 1: JPetStore ....................................................................................... 152
  8.3.1 Test Cases Design .......................................................................................... 153
  8.3.2 Experiment Results ....................................................................................... 154
8.4 Case Study 2: Pebble ............................................................................................ 166
  8.4.1 Test Cases Design .......................................................................................... 167
8.4.2 Experiment Results ........................................................................................................... 168
8.5 Case Study 3: ImageProcessor ......................................................................................... 175
  8.5.1 Test Cases Design ............................................................................................................ 176
  8.5.2 Experimental Results ..................................................................................................... 177
8.6 Threats to Validity .............................................................................................................. 186
8.7 Discussion ............................................................................................................................ 188
8.8 Summary ............................................................................................................................ 190

Chapter 9 Conclusions and Future Work ............................................................................. 191
  9.1 Key Contributions of the Research .................................................................................. 191
  9.2 Future Work ..................................................................................................................... 193

Reference .................................................................................................................................... 196
List of Figures

Figure 3.1 High-level Solution Development Process .................................................. 34
Figure 3.2 Overview of Performance and Energy Consumption Trade-off Analysis Tool .................................................................................................................. 39

Figure 4.1 Energy Consumption of JPetStore with System Configuration $C_1'$ and Different Workloads .................................................................................................. 50
Figure 4.2 Energy Consumption of JPetStore with System Configuration $C_2'$ and Different Workloads .................................................................................................. 50
Figure 4.3 Energy Consumption of JPetStore with Increased Packet Size and Different System Configurations ....................................................................................... 51

Figure 5.1 Performance and Energy Data Profiling Framework .................................. 57
Figure 5.2 Server Power Consumption with Different Workload ............................... 68
Figure 5.3 Energy Consumption per Task with Different Workload .......................... 68
Figure 5.4 Throughput with Different Workload .......................................................... 69
Figure 5.5 Energy Consumption per Task with HT on and off .................................. 69
Figure 5.6 Throughput with HT on and off ................................................................. 70
Figure 5.7 Energy Consumption per Task with Different VM Numbers ......... 71
Figure 5.8 Throughput with Different VM Numbers .................................................... 71
Figure 5.9 Energy Consumption per Task with Different Deployment Strategies .................................................................................................................. 72
Figure 5.10 Throughput with Different Deployment Strategies ................................. 72
Figure 5.11 Energy Consumption per Task with Different Resource Allocation Strategies .............................................................................................................. 73
Figure 5.12 Throughput with Different Resource Allocation Strategies ............... 73
Figure 5.13 Server Power Consumption with Different File Size................. 75
Figure 5.14 Server Memory Usage with Different File Size....................... 75
Figure 5.15 Energy Consumption per Task with Different File Size.............. 75
Figure 5.16 Throughput with Different File Size........................................ 75
Figure 5.17 Server Power Consumption with Different File Size.................. 77
Figure 5.18 Energy Consumption per Task with Different File Size.............. 78
Figure 5.19 Throughput with Different File Size........................................ 78
Figure 5.20 Energy Consumption with Different VM Configurations............. 79
Figure 5.21 Throughput with Different VM Configurations............................ 79
Figure 5.22 Energy Consumption per Task with Multiple Processes and Single VM................................................................. 80
Figure 5.23 Server Power Consumption with Multiple Processes and Single VM.......................................................................................... 80
Figure 5.24 Throughput with Multiple Processes and Single VM.................. 80
Figure 5.25 Energy Consumption with Multiple Processes and Multiple VMs. 81
Figure 5.26 Throughput with Multiple Processes and Multiple VMs............. 81
Figure 5.27 Energy Consumption per Task with Different VMs................... 82
Figure 5.28 Throughput with Different VMs................................................ 82
Figure 5.29 Energy Consumption per Task with Different Database Operation 84
Figure 5.30 Throughput with Different Database Operation......................... 84
Figure 5.31 Response Time with Different Database Operation.................... 84
Figure 5.32 Energy Consumption per Task with Different Record Sizes......... 85
Figure 5.33 Throughput with Different Record Sizes................................... 86
Figure 5.34 Response Time with Different Record Sizes............................... 86
Figure 5.35 Energy Consumption per Task with Different Packet Sizes........... 88
Figure 5.36 Throughput with Different Packet Sizes..................................... 88
Figure 5.37 Energy Consumption with Different VMs................................... 89
Figure 5.38 Throughput with Different VMs................................................ 89
Figure 5.39 Energy Consumption with Different Sizes of Data Block............ 90
Figure 5.40 Throughput with Different Sizes of Data Block ......................... 90
Figure 5.41 Energy Consumption with Different Deployment Strategies ........ 92
Figure 5.42 Throughput with Different Deployment Strategies ..................... 92
Figure 5.43 Energy Consumption with Different Numbers of User Requests ... 93
Figure 5.44 Throughput with Different Numbers of User Requests .............. 93
Figure 5.45 Energy Consumption with Different Deployment Strategies ........ 95
Figure 5.46 Throughput with Different Deployment Strategies ..................... 95

Figure 6.1 Simplified Stochastic Form Chart Meta-Model ......................... 105
Figure 6.2 High-level Class Diagram of Cloud Application Workload Mode . 107
Figure 6.3 Class Diagram of Cloud System Architecture Model ................. 108

Figure 7.1 StressCloud Performance and Energy Consumption Analysis Process ........................................................................................................ 120
Figure 7.2 A JPetStore High-level Workload Model in StressCloud .......... 125
Figure 7.3 Stochastic Form Chart Models of Computation-intensive Task (a),
  Data-intensive Task (b) and Communication-intensive Task (c) ............. 126
Figure 7.4 An Example Cloud Architecture Modelled in StressCloud ......... 128
Figure 7.5 An Example Workload Model Validation Result and Quick Fix ... 130
Figure 7.6 An Example Workload Model and Architecture Model for Energy
  Efficiency Validation in StressCloud ...................................................... 131
Figure 7.7 An Example Energy Efficiency Validation Result in StressCloud . 132
Figure 7.8 Development Scripts generated by StressCloud ...................... 133
Figure 7.9 Load Test Scripts ................................................................. 135
Figure 7.10 Visualised Performance (a) and Energy Data (b) .................... 137
Figure 7.11 High-level Architecture of StressCloud .............................. 138

Figure 8.1 Performance and Energy Consumption Profiling Framework with
  StressCloud ......................................................................................... 152
Figure 8.2 JPetStore High-level Workload Model ................................. 154
Figure 8.3 Energy Consumption with Different Numbers of User Requests... 155
Figure 8.4 Throughput with Different Numbers of User Requests .......... 156
Figure 8.5 Energy Consumption of MJPetStore with Different Deployment Strategies .................................................................................................................. 158
Figure 8.6 Throughput of MJPetStore with Different Deployment Strategies. 158
Figure 8.7 Energy Consumption of MJPetStore and RJPetStore with Different Deployment Strategies .................................................................................................................. 160
Figure 8.9 Energy Consumption with Different Numbers of User Requests... 162
Figure 8.11 Energy Consumption of MJPetStore with Different Deployment Strategies .................................................................................................................. 164
Figure 8.12 Throughput of MJPetStore with Different Deployment Strategies .................................................................................................................. 164
Figure 8.13 Energy Consumption of MJPetStore and RJPetStore with Different Deployment Strategies .................................................................................................................. 165
Figure 8.14 Throughput of MJPetStore and RJPetStore with Different Deployment Strategies .................................................................................................................. 166
Figure 8.15 High-level Workload Model for Pebble................................. 167
Figure 8.16 Energy Consumption with Different High-Level Workload Models .................................................................................................................. 170
Figure 8.17 Throughput with Different High-Level Workload Models........ 170
Figure 8.18 Energy Consumption of MPebble with Different File Sizes ....... 171
Figure 8.19 Throughput of MPebble with Different File Sizes................... 172
Figure 8.20 Energy Consumption of MPebble and RPebble with Different File Sizes .................................................................................................................. 172
Figure 8.21 Throughput of MPebble and RPebble with Different File Sizes .. 172
Figure 8.22 Energy Consumption of MPebble with Different Deployment Strategies .................................................................................................................. 174
Figure 8.23 Throughput of MPebble with Different Deployment Strategies... 174
Figure 8.24 Energy Consumption of MPebble and RPebble with Different Deployment Strategies .................................................................................................................. 175
List of Tables

Table 2.1 Cloud System Performance Evaluation Approaches.......................... 21
Table 4.1 Mixed Type of User Requests............................................................ 49
Table 4.2 Browsing Request Packet Size ........................................................... 51
Table 5.1 Specifications of HP Z400................................................................ 56
Table 5.2 Type of VM ........................................................................................ 57
Table 6.1 OCL-specified signature of Resource-Overcommitted Anti-Pattern110
Table 6.2 OCL-specified signature of Hyper-Threading Anti-Pattern............ 111
Table 6.3 OCL-specified signature of Memory-Usage Anti-Pattern ............... 112
Table 6.4 OCL-specified signature of Task-Parameter Pattern....................... 114
Table 6.5 OCL-specified signature of Task-Composition Anti-Pattern........... 115
Table 7.1 Background and Experience Level of Participants ....................... 142
Table 8.1 Types of Workloads ......................................................................... 168
Chapter 1

Introduction

This chapter provides an overview of this thesis. The background, motivation, key research issues and contributions of this research are introduced. First, an introduction of research background and motivation are presented in Section 1.1. Key research issues are introduced in Section 1.2. Then the key contributions of this research are given in Section 1.3. Finally, an overview of the structure of this thesis is provided in Section 1.4.

1.1 Research Background and Motivation

Cloud computing is a new and promising computing paradigm which delivers computing infrastructure as a utility [1]. It provides computation, software, data access, and storage services through the Internet like conventional utilities in everyday life (e.g. water, electricity, gas and telephony) [2].

The major downside of conventional in-house application deployment is the cost of the required acquisition and maintenance of hardware. For instance, Web application servers with a fixed capacity to handle the expected maximum application workload are needed, as well as the installation and maintenance of the entire software and hardware infrastructure of the platform supporting that application. Moreover, additional capacity may need to be purchased to handle any increased peak workload, even if this additional capacity is not used most of the
time. Such deployment and hosting approaches are expensive and wasteful. With the cloud model, the deployment and hosting of applications becomes cheaper and easier with the use of the flexible and elastic infrastructure services offered by cloud service providers. Cloud consumers are then able to pay service providers based on their usage of the services.

This approach allows consumers to: 1) achieve elastic computation and storage, i.e. the ability to dynamically scale those resources up and down according to real-time needs; 2) pay for only what they currently need; 3) avoid high up-front purchase and on-going infrastructure and service maintenance; 4) avoid in-house need of skills sets for special IT platforms; and 5) for small and medium enterprises (SMEs) in particular – leverage IT security, scalability, reliability and robustness that are difficult to obtain with purely in-house solutions. Therefore, more and more applications have been migrated to cloud environments. According to estimation in [3], the size of global cloud computing resources will reach $44.5 billion USD in 2016.

The proliferation of cloud computing has resulted in the establishment of large-scale data centres around the world containing thousands of servers. However, two major issues with the cloud model have been encountered: 1) the huge energy consumption of the required large-scale cloud data centres [4]; and 2) the need to meet ever-increasing system performance and other Quality of Service (QoS) requirements for cloud services [5]. High energy consumption directly contributes to data centres’ operational costs, especially as the energy unit cost continues to rise significantly [6]. Currently, power consumption contributes up to 42% of a data centre’s total expenses [7]. In addition, the huge amount of power consumption of data centres potentially accelerates global climate change. According to a New York Times study, data centres use about 30 billion watts of electricity per hour worldwide, equivalent to the output of 30 nuclear powerplants [8]. This is expected to continue to increase significantly. Thus, energy consumption has become a critical issue in modern cloud environments [9].

On the other hand, one of the key challenges for cloud service providers is to provide reliable QoS to end-users as they are extremely sensitive to the performance of the services. One of the most important QoS dimensions is the performance,
usually measured in throughput and response time. For instance, even a 100ms extra delay of cloud service response time can cause a 1% drop in sales [10]. Therefore, a key objective of cloud providers is thus to minimize energy consumption of cloud data centres while still guaranteeing increasingly strict Service Level Agreements (SLAs).

Many research efforts have been made to improve energy efficiency in cloud environments. Some simple techniques provide basic energy management for servers in cloud environments, i.e. turning on and off servers, putting them to sleep or using Dynamic Voltage/Frequency Scaling (DVFS) [11] to adjust servers’ power states. DVFS adjusts CPU power (and consequently the performance level) according to the workload. However, the scope of DVFS optimisation is limited to CPUs. Another approach for improving energy efficiency is to adopt virtualisation techniques to get better resource isolation and reduce infrastructure energy consumption through resource consolidation and live migration [12]. Using virtualisation techniques, several energy-aware resource allocation policies and scheduling algorithms have been proposed to optimise energy consumption in cloud environments [13].

However, energy consumption and system performance of cloud data centres varies greatly with different workloads [14] [15]. By its nature, the cloud workload is highly variable and application-specific. The large spectrum of applications running in cloud environments can exhibit diverse utilisation of different resources, resulting in different energy consumption and system performance. Thus, cloud application deployment and management solutions with minimum energy consumption whilst guaranteeing system performance and SLAs needs to be developed for cloud service providers in order to improve energy efficiency in cloud environment.

To achieve the abovementioned objective of cloud service providers, a thorough understanding of the architectural and performance characteristics, as well as energy consumption patterns, are required to reason about the most energy efficient deployment solution for cloud applications. However, finding the best cloud application architecture and deployment solution to optimize energy efficiency and system performance is an extremely challenging task, due to the complexity and
heterogeneity of cloud applications and deployment platforms [16]. Firstly, accurate system performance and energy consumption data of cloud applications need to be collected. Thus, an appropriate energy consumption model for cloud applications is required in the energy data collection process. Secondly, it requires the evaluation of system performance and energy consumption under various combinations of application workloads and platform configurations. Finally, trade-off analysis between system performance and energy consumption needs to be performed based on the collected data to find out the most energy-efficient cloud application deployment solution.

However, there are numerous different application workloads and platform configurations possible, even for a small application like JPetStore\(^1\). Generating performance test plans, changing system configurations, application configuration, and performance test plan configuration are tedious and error-prone. In order to address the abovementioned issue, a solution that is able to efficiently determine the best energy-efficient cloud application deployment strategy based on collected performance and energy data is imperative.

### 1.2 Key Research Issues

This thesis tackles the research challenges in relation to supporting performance and energy consumption analysis of cloud applications. In particular, the following research issues are investigated:

- **Energy Consumption Model:** Appropriate energy consumption measurements for cloud applications need to be collected before trade-off analysis. The granularity of the energy consumption calculation in most existing energy consumption models is a single hardware component without considering running applications in a cloud environment. Most energy models are based on the power model of each hardware component [4, 17-22]. Some of the energy models are based on the resource usage of Virtual Machines (VMs) and then converted to energy consumed based on the power model of each individual hardware resource [23, 24]. The impact

of diverse workloads and types of running applications in the cloud has to be taken into account. However, the energy consumption of cloud systems varies greatly with different workloads and various types of running applications in the cloud, as well as the cloud system resource configuration and allocation strategies [14]. Therefore, a new energy consumption model focusing on cloud applications needs to be designed.

- **Performance and Energy Consumption Trade-Off Analysis**: Prior to applying a new energy consumption model for energy consumption data collection, the model needs to be validated. Therefore, extensive experiments need to be conducted to profile and analyse the energy consumption of cloud applications based on the energy consumption model. In addition, the system performance of cloud applications also needs to be analysed in order to find out the trade-offs between performance and energy consumption.

- **Energy Consumption Signatures**: We aim to reduce the time and effort needed to determine the most energy efficient deployment strategy of cloud applications. Due to the variety of cloud system configurations and application workloads, it is time consuming to perform load tests and collect data in all the different scenarios. Therefore, it is necessary to provide guidelines to cloud application system architects and performance engineers about system performance and energy consumption under specific system configurations and workloads. Research efforts have been made to analyse the performance and energy consumption of cloud systems [15, 18, 23, 25]. However, no system performance and energy consumption patterns of cloud applications have been derived and formalised in order to improve the efficiency of trade-off analysis. Therefore, a series of system performance and energy consumption signatures of cloud applications needs to be developed.

- **Performance and Energy Consumption Analysis Tool for Cloud Applications**: Profiling and analyzing the system performance and energy consumption of cloud applications is time consuming. System architects and performance engineers need to run extensive experiments with different parameters, metrics and workloads. Manual generation of load test plans,
change of system configurations and application of load tests are very
tedious and error-prone. Therefore, a new performance and energy
consumption analysis tool that can automatically perform load test and
collect real performance and energy consumption data of cloud applications
needs to be developed. It should be able to accommodate different system
architectures and adopt different workload models during the trade-off
evaluation process.

1.3 Key Contributions

In this research we have introduced a set of solutions to accelerate the energy
efficiency evaluation process of cloud applications with different deployment
strategies and workload. Below we summarise the key contributions that we have
achieved:

1. *A novel energy consumption model for cloud applications* [26].
   We introduce a novel energy consumption model to profile and analyse the
   energy consumption of cloud applications. The energy consumption model
   provides a basis for characterising the energy consumption of cloud
   applications under different system configurations and application workloads.
   Instead of measuring the energy consumption of individual hardware
   components, a single task running in the cloud environment is considered as
   the fundamental unit for energy profiling in the energy consumption model.
   Types of cloud applications and their workloads, as well as system
   configurations have been taken into account in the energy consumption
   model.

2. *Experimental analysis of system performance and energy consumption of
   cloud applications* [27, 28].
   We analyse the results from experiments that we have conducted to profile
   and investigate system performance and energy consumption of different
   types of cloud applications with varying workloads and system
   configurations. We investigate the energy efficiency of single cloud
   application, as well as multiple cloud applications running one the same
server/VM simultaneously. The experimental results show the correlation coefficients of energy consumption, system configurations and workloads, as well as the performance of the cloud applications. In addition, a set of guidelines are derived from the experimental results. These guidelines can be adopted to achieve energy efficient deployment, resource provisioning and management strategies for cloud applications.

3. A set of novel, formalised, extensible energy consumption signatures for cloud application.

We extracted a series of energy consumption patterns from our experimental results which are formalised as “signatures” using the Object Constraint Language (OCL). In order to support formalise these energy consumption patterns, we have developed: 1) a system performance and energy consumption pattern definition schema to capture the details of a given system performance and energy consumption evaluation scenario, including cloud application workload, cloud system configurations, required resources, application deployment strategies; 2) a cloud system architecture model to describe the architectures and system configurations of cloud systems; and 3) a workload model to capture the details of cloud applications’ workloads. System architects and performance engineers can adopt these energy consumption patterns to predict the energy efficiency of a target cloud application without running load tests. These patterns can be applied to different cloud applications in different cloud environments as they are generic and independent with cloud platforms.

4. A novel, automatic, system performance and energy consumption analysis tool for cloud applications – StressCloud [29].

We have developed StressCloud, an automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments. Key novel contributions of StressCloud are: 1) supporting user-defined high-level architecture and workload models for complex cloud applications; 2) supporting energy efficiency validation of complex cloud applications using formalised “signatures” based on high-level architecture and workload models; 3) fully automatic generation and deployment of large-scale cloud application workload test services and cloud application
model prototype implementations; 4) ability to realistically energy and performance stress test existing cloud applications and potential cloud application models; 5) automatic profiling of system performance and energy consumption of cloud applications; and 6) analytical support for pre-test model energy and performance weaknesses and post-test energy and performance metric analysis.

5. Extensive validation of StressCloud on exemplar cloud applications, models and workloads.

We have conducted three case studies to further validate StressCloud and the energy consumption signatures. We investigated the energy efficiency of different types of cloud applications for both modeled applications and the corresponding real applications. The evaluation results have validated the practical value of StressCloud approach.

1.4 Overview of Thesis

This thesis addresses the problem of efficiently determining the best energy-efficient cloud application deployment strategy based on collected performance and energy consumption data.

Chapter 2 reviews related work. It introduces the important concepts used in this research, including cloud computing, cloud system performance evaluation, and approaches to achieve energy efficiency in cloud systems. The research problems are described and discussed in detail.

Chapter 3 presents our approach to the addressing the key research issues identified in Section 1.2. We first introduce our novel energy consumption model for cloud applications. Second, we describe our approach to trade-off analysis between performance and energy consumption. Third, our approach to energy consumption signatures is presented. Finally, we discuss our development of a new cloud application performance and energy consumption analysis tool.

Chapter 4 introduces our energy consumption model for cloud applications. We first present the details of this novel energy consumption model. Then we present an
exemplar usage of the energy consumption model using an example. Finally, we
discuss the key advantages and limitations of this energy consumption model.

Chapter 5 presents empirical experiments used to analyse the performance and
energy consumption of various cloud applications. First, the experimental setup is
described. Second, the detailed test cases designs of each cloud application are
presented. Third, the experimental results are presented and discussed in detail.
Finally, we discuss the findings we derived from the experimental results.

Chapter 6 describes the energy consumption signatures of cloud applications
derived from the experimental results in Chapter 5. In particular, we first introduce
the formalised energy efficiency pattern definition schema. Then we present each
pattern and its signature specification in detail. Finally, we discuss the key
advantages and limitations of these new energy consumption signatures.

Chapter 7 introduces StressCloud – an automatic performance and energy
consumption analysis tool for cloud applications. We first provide an overview of
StressCloud’s capabilities and the energy and cloud application performance
analysis approaches applied in StressCloud. Then we illustrate the usage of
StressCloud applied to an exemplar problem. The details of the tool design and
implementation are also presented. Then we describe an evaluation of StressCloud
and present and discuss the evaluation results. Finally, we discuss the key
advantages and limitations of StressCloud.

Chapter 8 presents three case studies of using StressCloud to analysis system
performance and energy consumption. We introduce a set of case studies to 1)
further validate StressCloud; 2) further validate the energy consumption signatures;
and 3) show how we use StressCloud to further analyse the trade-off between
system performance and energy consumption of different cloud applications. We
first give a brief introduction to the selected cloud application. Then for each case
study, we 1) present the test case design; and 2) analyse the experimental results.
Finally, we discuss the conclusions drawn from the experimental results analysis.

Chapter 9 summarises this thesis, the major contributions of this research and
discusses the future research directions.
Chapter 2

Literature Review

In this chapter, we review and summarise key existing research efforts and the state-of-art research results related to this thesis, including cloud computing, cloud system performance evaluation, and energy efficiency in cloud systems.

This chapter is organised as follows. In Section 2.1, we give an overview of the research areas covered in our state-of-the-art analysis. In Section 2.2, we introduce the concepts of cloud computing. In Section 2.3, we review the related work in the area of cloud system performance evaluation. In Section 2.4, we review the work related to energy efficiency in cloud systems, including energy consumption model, energy saving policies, and energy profiling and analysing. Finally, we summarise this chapter in Section 2.5.

2.1 Introduction

The main research objective of this thesis is to find solutions which can efficiently determine the energy efficient deployment strategy of cloud applications. To help achieve this research objective, a number of related research areas need to be studied.

We have determined three main relevant research areas. Below we discuss the key points and questions that we need to get answers for in each research area in order to specify what existing research outcomes and their limitations or what efforts could help achieve our research objective:

- **Cloud Computing**: We need to study the cloud computing model details and determine what architectural characteristics contribute to the performance of cloud systems. What are the key performance requirements that should be
addressed when deploying a cloud application? What is the current research state of energy consumption of cloud systems?

- **Cloud System Performance Evaluation**: We need to study what the existing software performance evaluation approaches are. What are the main advantages and limitations of these approaches? How are these approaches extensible to support evaluating the performance of existing cloud applications as well as what-if analysis of new designed cloud applications? What are the main performance evaluation tasks? How much automation do these approaches provide to facilitate the automated performance evaluation of cloud systems?

- **Energy Efficiency in Cloud Systems**: What is the current status of energy consumption in cloud environments? What are the causes and problems of high energy consumption? What are the existing efforts which have been made to cloud system energy consumption analysis or energy saving? What do energy consumption models of cloud systems exist? What are the key limitations of those existing models? What are the existing energy consumption patterns analysis efforts? What are the limitations of these efforts that arise from applying these techniques to cloud application energy efficiency analysis?

The abovementioned three main relevant research areas are discussed in Section 2.2, Section 2.3, and Section 2.4, respectively.

### 2.2 Cloud Computing

Cloud computing is a new and promising computing paradigm which provides rented services for computation, application software, and data storage via the Internet [30]. It provides IT enterprises with a flexible and easy way to operate, manage and maintain their own IT business assets, such as servers, networks, data and applications. Compared to a traditional in-house computing model, cloud computing offers several key advantages, including on-demand scaling, resource multiplexing, pay-as-you-go metred service, and high-speed network access [31]. These advantages of cloud computing have encouraged more and more IT companies to move their businesses to the cloud, especially for SMEs (Small and
Medium Enterprises) – leverage IT security, scalability, reliability and robustness that are difficult to obtain with purely in-house solutions.

Cloud computing has a service oriented architecture in which services are provided at three different levels - *Infrastructure as a Service* (*IaaS*), *Platform as a Service* (*PaaS*), and *Software as a Service* (*SaaS*) [32]. *IaaS* provides basic infrastructure components such as CPUs, memory, and storage. Amazon’s Elastic Compute Cloud (EC2)\(^2\) is a prominent example for an *IaaS* offer. On top of *IaaS*, more platform-oriented services allow the usage of hosting environments tailored to a specific need. Google App Engine\(^3\) is an example for a Web *PaaS* which enables to deploy and dynamically scale Web applications. Finally, the top-most layer provides its users with ready-to-use applications also known as *SaaS*. It provides end-users access to business functionality remotely (usually over the Internet) as a service. Leading companies in IT industry are gradually moving their applications and related data to the cloud and delivering them as services [33]. For example, Microsoft is offering their development and database services through Microsoft Azure\(^4\).

The flexibility and scalability of commercial cloud infrastructure makes it an attractive application deployment target. In cloud environments, the applications are developed as independent sets of interacting service units, which perform functions from simple requests to complicated business processes [34]. Before deploying applications to the cloud, service providers need to make sure that their applications can meet end-users Quality of Service (QoS) requirements. End-user of cloud applications are extremely sensitive to the performance of the services, usually measured in throughput and response time. For instance, even a 100ms extra delay of cloud services response time can cause a 1% drop in sales [10].

When deploying an application in the cloud, there are two major tasks: 1) select a number of cloud resources (e.g. servers or virtual machines which can meet the QoS requirements) to run the cloud application; and 2) determine how to deploy the service units of the cloud application on the selected cloud resources. However, due to the associated complexity of cloud environments, it is difficult to determine

\(^2\) http://aws.amazon.com/ec2/
\(^3\) https://cloud.google.com/appengine/
\(^4\) https://azure.microsoft.com/en-us/
deployment strategies, especially for newly migrated applications [35]. For example, while cloud environments are good candidates for supplementary system platforms during occasional overload of Internet applications (e.g., electronic commerce), reports on Amazon EC2 consistently mention that network latency may affect overall system performance considerably [36-38].

In addition, the proliferation of cloud computing has resulted in the establishment of large-scale data centres around the world containing thousands of servers. Cloud data centres consume huge amount of electrical energy resulting in high operational costs and CO₂ emissions to the environment. The power drawn by data centres ranges from a few kilowatts (KW) for a rack of servers to several tens of megawatts (MW) for large facilities. For the higher power density facilities, electricity costs are a dominant operating expense and account for over 10% of the total cost of ownership (TCO) of a data centre [39]. Another problem of high power consumption and increasing density of server components is the heat dissipation. Overheating of the components can lead to decrease of their lifetime and high error-proneness. Therefore, power is required to feed the cooling system operation.

On the other side, for cloud service providers, they need to provide reliable services to cloud users with satisfactory system performance, such as throughput and response time. Thus, a key and common objective of cloud service providers is thus to develop cloud application deployment and management solutions with minimum energy consumption while guaranteeing performance and other QoS specified in Service Level Agreements (SLAs). Therefore, before deploying an application in the cloud, it is important to understand its performance and energy consumption characteristics when subject to different architectural decisions and deployment strategies in cloud environments.

2.3 Cloud System Performance Evaluation

Software architecture plays a large role in the software performance evaluation process during the software design phase [40]. However, it is difficult for system architects to determine the appropriate architectural organisation that will meet performance requirements during architectural design for cloud applications. In
order to help architects get a more accurate assessment of their architectural design, various approaches have been proposed to validate architectural designs, including 1) performance monitoring of existing systems [41]; 2) simulation and modelling [42, 43]; and 3) model-based performance test-beds [44].

A lot of research efforts have been made in evaluating cloud system performance by researchers based on the abovementioned three approaches. This section discusses the existing efforts and their key advantages and limitations. The details of the performance evaluation approaches are presented in Section 2.3.1, Section 2.3.2 and Section 2.3.3, respectively.

2.3.1 Performance Monitoring

In order to guarantee the performance required by cloud applications, software architects have to 1) quantify capacity and resources (e.g. CPU, memory, storage, etc.) required by the application, depending on different architectural choices, and 2) determine the estimated workload. Performance monitoring is a key tool which provides information and Key Performance Indicators (KPIs) for cloud systems and helps cloud architects to achieve the abovementioned goals. By monitoring existing cloud applications, architectural design and resource usages of these applications can be summarised and analysed if the characteristics of the application deployment environment can be realised. Performance monitoring allows service providers to implement mechanisms to prevent or recover SLA violations in existing cloud systems. In addition, the performance monitoring results can be used as benchmarks when deploying newly designed cloud systems. Based on the analytical results, an architectural style that can provide better performance and/or availability than others will be chosen.

Extensive research efforts have been made in monitoring cloud system performance. Most of the existing efforts contain a module specifically targeted to performance monitoring which are addressed as follows.

CloudStone [45] is a UC Berkeley project aimed at providing a benchmark for reproducible and fair performance assessment of cloud applications. It specifies a database and workload for studying cloud infrastructures and defines performance
metrics to compare alternative cloud application frameworks and configurations. A set of automation tools is provided for generating load and monitoring its performance of different deployment options. Faban\(^5\), an open source Markov-chain based workload generator, is adopted to generate workloads during the system performance evaluation process. It supports Amazon EC2 as the application deployment infrastructure.

OpenNebula [46] is a toolkit for the management of distributed and heterogeneous public, private and hybrid cloud infrastructures. Through a module called Information Manager, it monitors cloud physical infrastructure and provides information of the system performance data to cloud providers. Monitoring data are collected through probes installed on the physical servers, queried through SSH connections, and they are related to information concerning to the status of physical servers.

Aneka [47] is a framework for development, deployment and management of cloud applications. It consists of a scalable cloud middleware that is deployed on top of cloud resources and extensible collection of services, coordinating the execution of applications and monitoring the status of the cloud. Aneka provides an extensible API for the development of cloud applications and supports both public and private clouds. The framework includes the basic services for cloud resources management, application execution and system monitoring.

This kind of performance evaluation approach – performance monitoring - requires close similarity between the existing application and the proposed application. Very often, considerable modification of the proposed application is required in order to gain any useful performance results. In addition, this approach is limited to provide benchmark style guidelines only for architects.

2.3.2 Simulation-based Modelling

By modelling the interactions and behaviours of each component based on the architecture of proposed cloud system, the whole system can be simulated. Simulation modelling is steadily becoming more practical with the availability of more powerful inexpensive computers. Therefore, it is becoming the most popular

\(^5\) http://faban.sunsorce.net
methodology to conduct performance evaluation or prediction of new software systems. A number of cloud system performance evaluation tools has been developed by researchers using simulation-based modelling approach which are addressed as follows.

CloudSim [48] is a simulation toolkit for evaluating the performance of Cloud computing infrastructure. It is a self-contained platform that can support modelling and simulation of cloud infrastructures containing data centres, users, user workloads and pricing models. It provides a virtualisation engine with extensive features for modelling life-cycle management of virtual machines in a data centre, including policies for provisioning of virtual machines to hosts, scheduling of resources of hosts among virtual machines, scheduling of tasks in virtual machines and modelling of costs incurring in such operations. An entity named Utilisation Model exposes methods and variables for defining the resource and VM-level requirements of a cloud application at the instance of deployment. However, the users need to extend the Utilisation Model to build their own workload models.

SPECI [49] is a simulation tool that enables exploration of scaling as well as performance properties of large data centres in cloud environments. This tool enables simulation of the state of near permanent hardware failures and load balancing of virtual machines in a data centre. Based on these simulations, this tool can evaluate the behaviours and performance of the data centres with reproducible results, prior to designing and building the data centres. It is focused on modelling of the behaviours of cloud server side without considering the workload of cloud applications.

CloudAnalyst [50] is a simulation tool built directly on top of CloudSim [48]. It enables application developers or designers to determine the deployment strategy which has better performance for allocation of resource among available data centres in cloud environments. Furthermore, it allows descriptions of application workloads, including information of geographic location of users and data centres, number of users, number of data centres and number of resources in each data centres. Based on these descriptions, CloudAnalyst can generate information about response time of request, processing time of requests and other performance metrics.
iCanCloud [51] is a simulator for large cloud data centres. It is designed based on SIMCAN [52], which is a software simulation framework for large storage networks. It can predict the trade-off between costs and performance of a particular application in a specific hardware in order to inform the users about the costs involved. It provides an API and an adapted MPI library for modelling and simulating applications by using traces of real-world applications or a state graph, but existing cloud application can only be modelled manually. It also allows parallel execution of one load test over several machines.

EMUSIM [53] combines emulation and simulation to extract information automatically from the application behaviour via emulation and uses this information to generate the corresponding application simulation model. Such a simulation model is then used to build a simulated scenario that is closer to the actual target production environment in application computing resources and request patterns.

CloudProphet [54] is a trace-and-reply tool to predict a legacy application’s performance if migrated to a cloud infrastructure. It uses architecture independent characteristics extracted from workload traces to find the most similar performance benchmarks of legacy applications. It focuses on predicting the performance of CPU-intensive applications across a large collection of CPU types.

The major limitation of abovementioned cloud system performance evaluation tools is that they cannot be applied in analysing the energy efficiency of the target cloud system. However, some of the performance evaluation tools have taken into account the energy consumption of data centres which are addressed as follows.

GreenCloud [55] is a simulation environment for energy-aware cloud computing data centres. It is designed to capture details of the energy consumed by data centre components (e.g., servers, switches, and links) as well as packet-level communication patterns in realistic setups. GreenCloud is developed as an extension of a packet-level discrete-event network simulator NS-2 [56]. The simulation results obtained for two-tier, three-tier, and three-tier high-speed data centre architectures demonstrate the effectiveness of the simulator in utilising power management
schema, such as voltage scaling, frequency scaling, and dynamic shutdown that are applied to the computing and networking components.

MDCSim [57] is an event-driven simulation platform which focuses on layered data centre architectures and cluster configuration. It can analyse a cluster-based data centre with detailed implementation of each individual tier. It has been configured into three layers, including a communication layer, a kernel layer and user-level layer, for modelling the different aspects of a cloud, and can estimate the throughput, response times, and power consumption. The latter is approximated using linear functions of the server utilisation, which in turn is calculated based on the number of nodes, number of requests and average execution time of requests.

Green Data Centre Simulator (GDCSim) [58] is a simulation tool that unifies existing techniques to green data centre management and allows holistic physical data centre design and analysis before deployment. Specifically, GDCSim allows analysing data centre energy efficiency by studying and testing: 1) different data centre geometries; 2) workload characteristics; 3) platform power management schemes; 4) scheduling algorithms; and 5) data centre configurations. The functionality of GDCSim is demonstrated by two case studies with two different data centre layout and workload types. However, it is especially indicated to study the design of green data centres without considering the performance aspects of the data centres.

However, the major limitation of a simulation-based modelling approach is that it is very difficult to obtain performance models and system behaviours, especially for third party applications such as databases. Compared to system development itself, simulation models can sometimes take nearly as long. Moreover, any incorrect environment settings have the potential to alter the results of the simulation and give the wrong results. Therefore, simulation results are more or less inaccurate and have limited usefulness because of its very nature [43].

2.3.3 Test-beds

A few of the performance evaluation approaches discussed in the previous section can be used to analyse the energy efficiency of the target cloud system. However,
they are using simulation techniques. While simulation offers a number of advantages, especially in terms of scalability and experiment repeatability, it is still based on assumptions and simplifications that might not fully represent an actual cloud system. For this reason, it may be preferable to use a real-world cloud environment to investigate the energy efficiency of cloud systems.

For example, in [59], Srikantaiah et al. used a private cloud of four physical machines to measure power consumption against CPU and disk utilisation. They used the Xperf utility to monitor resource utilisation and the WattsUp\(^6\) Pro ES power metre to measure power consumption. Through their experiments they identified empirically optimal points for CPU, disk and energy values. Then they applied the same configuration to test a consolidation algorithm. This algorithm allocates incoming workload based on the Euclidean distances that a workload allocation would have if assigned to a specific server with the optimal values that were empirically obtained. They found that energy used by the proposed heuristic was about 5.4% more than optimal points on average.

Van et al. [60] build a cloud environment of three physical machines of four CPU cores each, running one virtual machine per core or a group of two virtual machines per two cores. They generated a series of applications with different priorities to validate the resource arbitration of heterogeneous applications and show that the balancing of the quality of service and energy can be achieved by prioritising the applications.

Performance test results in real-world cloud environments are more accurate. However, there are no unified supporting tools to generate cloud workloads and collect performance and energy data. Therefore, the key limitation of the abovementioned approaches is that they can only be applied to analyse the energy efficiency of existing cloud systems. However, before deploying a cloud system, a what-if analysis is required to obtain the most energy efficiency deployment strategy. In order to address this issue, a model-based performance test-bed is needed to help evaluate the energy efficiency of both existing and newly designed cloud system in real-world cloud environments. A model-based performance test-bed is a platform for performance testing and analysis of software systems,

\(^6\) [https://www.wattsupmeters.com/secure/index.php](https://www.wattsupmeters.com/secure/index.php)
which is generated by modelling system behaviours. This approach offers many advantages of tool developing, including increased productivity, enhanced source code consistency, and high-level reusability. However, it requires more resources such as hardware comparing to other performance prediction methodologies, i.e. performance monitoring and simulation. Based on our best knowledge, no model-based performance test-bed has been built for cloud systems.

2.3.4 Discussion

Table 2.1 shows a comparison between key aspects of previously discussed performance evaluation tools, and identifies whether they are used for investigating energy efficiency or performance in cloud environments. It also specifies whether these performance evaluation tools are monitoring systems, simulators or test-beds. A tick symbol is used in Table 2.1 to indicate what attributes the evaluation tools have.

As presented in Table 2.1, only a few of the performance evaluation approaches have taken into account the energy consumption of cloud systems. However, most of the energy efficiency evaluation approaches are built based on simulation-based modelling. As we discussed in Section 2.3.2, the test results of simulation-based modelling approaches may be inaccurate because of the imperfection of environmental configuration and input data in the simulation. In contrast to simulation-based modelling, model based test-bed generation provides more accurate test results because a test-bed is a more realistic representation of the real-world software environment. Compared to simulation, less work has been done in generating a model-based performance and energy test-bed for cloud systems.
Table 2.1 Cloud System Performance Evaluation Approaches

<table>
<thead>
<tr>
<th></th>
<th>Energy Efficiency</th>
<th>Performance</th>
<th>Monitor</th>
<th>Simulator</th>
<th>Test-bed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloudStone [45]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenNebula [46]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aneka [47]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CloudSim [48]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SPECI [49]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CloudAnalyst [50]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>iCanCloud [51]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>EMUSIM [53]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CloudProphet [54]</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>GreenCloud [55]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MDCSim [57]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>GDCSim [58]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Van et al. [60]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Srikantaiah et al. [61]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.4 Energy Efficiency in Cloud Systems

Traditionally, the design of cloud systems has focused on performance improvements driven by the demand of cloud consumers. However, energy consumption has become a critical issue in cloud systems as higher energy consumption will result in both high CO₂ emissions and high operational costs. The operation of large data centres which consists of a large number of cloud servers requires considerable amount of energy that accounts for a large slice of the total operational costs of cloud data centres [62]. The Gartner group estimates energy consumptions to account for up to 10% of the current data centre operational expenses, and this estimation may rise to 50% in the next few years [63]. High power consumption of cloud servers also generates heat and requires an accompanying cooling system that costs a rage of $2 million to $5 million per year for classical data centres [64]. Apart from high operation costs, high energy consumption results in substantial CO₂ emissions, which were estimated to be 2% of the global emissions [65]. Therefore, the reduction of energy consumption has become a first-order objective in the design of cloud systems.

Many research efforts have been made to improve energy efficiency in cloud environments. In this section, we give an overview of the recent research outcomes in improving energy efficiency of cloud systems, and discuss the open research challenges and limitations. Section 2.4.1 introduces the power and energy consumption models of cloud systems. Section 2.4.2 presents the existing energy saving policies. Section 2.4.3 discusses the energy profiling and analysing approaches in cloud environments.

2.4.1 Power and Energy Consumption Models

To understand power and energy management mechanisms, it is essential to clarify the terminology. Power and energy can be defined in terms of a work that a system performs. Power is the rate at which a system performs the work, while energy is the total amount of the work performed over a period of time. Power and energy are measured in watts and joule, respectively.
\[ E = P \times T \] (2.1)

Where \( P \) is power, \( T \) is a period of time, and \( E \) is energy. The difference between power and energy is important since a reduction of power consumption does not always reduce the energy consumed. For instance, power consumption of a cloud application can be decreased by reducing the number of physical servers. However, in this case, the workload of the cloud application may take longer to complete resulting in higher energy consumption.

In order to evaluate the energy consumption of cloud systems and understand its impact, it is necessary to create a model of cloud systems power/energy consumption. Such a model should be able to capture the characteristics that are related to the energy consumption of cloud systems.

The main power consumed by a cloud server is accounted for the CPU, followed by the memory and losses due to an inefficient power supply [66]. CPUs can consume less than 30% of their peak power in low-activity modes and more than 70% dynamic power ranges of the peak power. Therefore, research efforts have been made to investigate the power model of CPUs.

Fan et al. [62] found a strong relationship between the CPU utilisation and total power consumption by a server. The idea behind the proposed model is that power consumption by a server grows linearly with the growth of the CPU utilisation from the value of power consumption in the idle state up to the power consumed when the server is fully utilised. This relationship is shown in Equation (2.2):

\[ P(u) = P_{\text{idle}} + (P_{\text{busy}} - P_{\text{idle}})u \] (2.2)

where \( P \) is the estimated power consumption, \( P_{\text{idle}} \) is power consumption by an idle server, \( P_{\text{busy}} \) is the power consumed by the server when it is fully utilised, and \( u \) is the current CPU utilisation. The authors also proposed a nonlinear model as presented in Equation (2.3):

\[ P(u) = P_{\text{idle}} + (P_{\text{busy}} - P_{\text{idle}})(2u - u') \] (2.3)

where \( r \) is a calibration parameter that minimises the square error and has to be obtained experimentally. The authors conducted extensive experiments on thousands of nodes under different types of workloads and showed that the derived models
accurately predict power consumption by server systems with the error below 5% for the linear model and 1% for the empirical model.

The linear model and empirical model have been widely used to develop energy aware cloud management mechanisms. Based on the linear model, Lee and Zomaya [4] proposed an energy model of cloud tasks for developing energy-conscious task consolidation algorithms that reduce energy consumption of cloud systems. Instead of focusing on the server power consumption, the authors claimed that the energy consumed by a specific task is linear to the resource utilisation of the task. Jung et al. [21] proposed the Mistral system, which adopts the empirical model expressed in Equation (2.3) to predict server power consumption. Based on the resource utility and server power consumption, Mistral optimises power consumption of resources and performance of applications at data centre physical server level.

Dhiman et al. [67] found that although regression models based on just the CPU utilisation are able to provide reasonable prediction accuracy for CPU-intensive workloads, they tend to be considerably inaccurate for the prediction of power consumption caused by I/O- and memory-intensive applications. The authors proposed a power modelling methodology based on Gaussian mixture models that predicts power consumption by a physical machine running multiple virtual machine (VM) instances. To perform predictions, in addition to CPU utilisation, the model relies on run-time workload characteristics such as the number of instructions per cycle (IPC) and the number of memory accesses per cycle (MPC).

Based on the power or energy consumption models discussed above, the system power or energy consumption can be predicted at a certain level. The prediction results can be used as guidelines to prevent allocating unnecessary power to cloud servers. Thus, the total power or energy consumption can be saved. One major limitation of these power and energy models is that they only focus on single hardware components of cloud servers, such as CPU, memory, disk, etc. In reality, energy consumption of cloud servers is also related to system resource configurations and the workloads in the cloud. By its nature, these cloud workloads are highly variable and application specific. Thus, the energy consumption patterns of cloud applications should be taken into account. However, none of the existing
power or energy consumption models can characterise the energy consumption of cloud applications.

2.4.2 Energy Saving Policies

The power consumption of cloud systems depends on the types of hardware components and specific usage scenarios of these hardware components. The power consumption of cloud systems can be reduced by 1) applying hardware level energy saving techniques to specific hardware component, such as CPU, disk, network devices, etc.; and 2) applying system level energy saving mechanism to dynamically change system management strategies and configurations. Researchers have devoted their efforts to develop both hardware level and system level power management techniques to reduce power consumption in cloud systems.

- Hardware Level

A lot of energy saving methods have been proposed at the hardware level. Dynamic Voltage and Frequency Scaling (DVFS) [68] has been widely used for saving the power consumption of CPUs. The main idea of this technique is to intentionally scale down the CPU performance, when it is not fully utilised, by decreasing the voltage and frequency of the CPU. Therefore, the power consumption of the CPU decreases.

In addition, DVFS cannot only decrease the power of the CPU, but also save more energy to cool down the CPUs. Poess et al. [69] had employed in-memory technology to improve the power efficiency of data centres. The main idea of in-memory technology is to hold as much data set as possible in memory for data processing systems. They evaluated the trade-off between total power consumption and performance by considering multiple storage options for the database tables with different cost characteristics. They found that the in-memory configuration offers the best performance and consumes the least amount of power among all the tested configurations.

Carrera et al. [70] proposed an approach to save disk energy for Web servers by using multi-speed disks, such that each disk is slowed down for lower power consumption during periods of light load. Multi-speed provides consistent benefits
for realistic parameters. In particular, the disk energy saving produced by a
two-speed disk range from 14%-23%, compared to a similar conventional disk,
without any noticeable degradation in server performance. Similar approaches have
also been proposed in [71-75].

- **System Level**

  Many efforts have been made to improve the energy efficiency of cloud systems
  at system level. These approaches mainly utilise the knowledge of real-time resource
  usage and application workloads to optimise energy consumption.

  A simple but useful approach for saving energy consumption of cloud servers is
  simply turning them off in the idle time [76]. The main benefit of this approach is
  that it can be used in almost all hardware devices in cloud environments, not only
  for cloud servers. However, mechanisms for predicting the upcoming idle period of
  cloud devices need to be developed based on the characteristics of system events.

  Another approach for improving the utilisation of cloud server resources, and
  thus, reduce power consumption, is virtualisation of computing resources [77].
  Virtualisation introduces an abstraction layer between an Operation System (OS)
  and hardware. Physical resources can be split into a number of logical slices called
  Virtual Machines (VMs). Each VM can accommodate an individual OS creating for
  the user a view of a dedicated resource and ensuring the performance and failure
  isolation between VMs sharing a single physical machine. Virtualisation allows one
  to create several VMs on one physical server. Therefore, the amount of hardware in
  use can be reduces and the utilisation of resources can be improved accordingly.
  Based on virtualisation techniques, several energy-aware resource allocation policies
  and scheduling algorithms have been proposed to optimise energy consumption in
  cloud systems. One key benefit of virtualisation is the ability to easily move VMs
  from one physical host to another using live or offline migration. Beloglazov et al.
  [78] proposed a management policy for live migration of VMs, while still ensuring
  reliable QoS. The main idea is to migrate VMs to other live servers to free the
  current server, and switch the idle node to sleep mode based on current resource
  utilisation. Thus, energy consumption can be minimised.
Adopting energy-aware resource allocation and management mechanisms can also improve energy efficiency. Chase et al. [79] studied the problem of managing server resources in Internet hosting data centres. Servers are shared among multiple service applications with SLAs defined in terms of throughput and latency constraints. The authors developed Muse, an OS for an Internet hosting centre aimed at managing and coordinating interactions between a data centre’s components. Muse helps achieve energy efficiency by automatically scaling back the power demand by servers.

Chen et al. [80] proposed an approach to managing multiple server applications in hosting centres for minimising energy consumption, while meeting SLA requirements. The approach consists of two base phases executed periodically: 1) using queuing theory based approach to predict the workload for the near future and allocating a number of servers to each application to serve the predicted workload; and 2) using feedback-based control theoretic approach to set the DVFS parameters on servers suitable for serving the corresponding application’s workload. After each server allocation phase, the servers becoming idle get switched off to conserve the energy.

Verma et al. [81] used the characteristics of VMs, such as cache footprint and the set of applications running on the VMs, to derive power-aware placement of VMs. The authors investigated modelling the power consumption of High Performance Computing (HPC) applications and the impact of platform virtualisation. The power model was used to design a power-aware application placement controller.

The abovementioned works have made some initial efforts to develop energy saving policies at both hardware level and software level. However, none of them has identified the relationship between energy consumption and cloud applications with different configurations as well as system performance. Moreover, most of the research efforts can only be applied in existing cloud systems. However, it is critical to perform what-if energy efficiency analysis before real deployment.
2.4.3 Energy Profiling and Analysis

Prior to developing energy saving policies or mechanisms of cloud systems, it is essential to understand the energy consumption of cloud servers as they are the dominating part of energy consumption in cloud systems. Energy data profiling and analysis provide a vehicle for understanding the cloud-wide power and energy usage and self-managing cloud resources for energy efficiency and assurance.

CPU, memory, and disk are major components which contribute to server power consumption [66]. It is important to investigate the relationship between these components and the power consumption of cloud servers. Therefore, research efforts have been devoted to profiling and analysing the power consumption of cloud server hardware components.

Chen et al. [18] profiled and analysed the power consumption of a cloud server with different high performance computing workloads. The authors profiled the power consumption of CPU, memory and disk with different computation-intensive workload. Based on the profiling results, they built a linear power model that represents the behaviours of a single cloud server and includes the contribution from individual components, i.e. CPU, memory and HDD, to the total power consumption of a single cloud server.

The abovementioned profiling and analysis approach focused on the hardware components of cloud servers. However, this kind of approach does not provide a macroscopic view of the power consumption of the entire cloud system. The impact of virtualisation and software components on the power dynamics is also important in cloud environments.

Kansal et al. [23] proposed a solution for VM power metering, named Joulemeter. Joulemeter measures power consumption at the coarse level of individual VMs running on a physical server. It uses the same power management mechanisms that have proven beneficial on physical servers, i.e. hardware resource usages. VM power can be metred by tracking each hardware resource used by a VM and converting the resource usage to power usage based on a practical power model for the resource. The major practical power models of the resources including CPU, memory and disk. By using Joulemeter, the power consumption of VMs can be
measured without any additional equipment. However, it cannot model the power usage of other resources such as Ethernet cards.

Ge et al. [82] proposed a formwork named PowerPack for power profiling and evaluation of multicore, multiprocessor-based distributed systems and applications. The authors used PowerPack to profile the power consumption of parallel scientific applications and investigate the application power behaviour. They insert a set of user-level APIs before and after the code region in the interest to map the power profile to the source code of the application.

Govindan et al. [83] proposed a power provisioning technique which exploits knowledge of the power usage patterns of hosted applications, named power profiles. The authors profiled a set of applications to illustrate the process of deriving an application’s power consumption behaviours. Based on the statistical properties retrieved from the power profiles, power provision framework was built to provision power needs of hosted workloads and their aggregates. The main idea is to reduce power over-provisioning while keeping the risk of significant economic penalties resulting from reduced performance below acceptable levels.

Zhang and Fu [14] profiled power and energy at server granularity to characterising the energy consumption of cloud systems. They collected the power and energy usage data to analyse with different system configurations, including 1) overhead in power and energy usage by virtualisation; 2) CPU configurations and power usage; 3) memory configurations and power usage; and 4) disk configurations and power usage.

Feng et al. [84] studied power-performance efficiency of scientific applications. The authors not only profiled energy efficiency of hardware components, i.e. CPU, memory, disk and Network Interface Controller (NIC), they also profiled energy efficiency of physical servers. They also investigated the energy efficiency when running the scientific application on different numbers of physical servers.

Metri et al. [15] profiled the energy consumption with different types of applications running on heterogeneous cloud servers. The authors proposed Application Specific Energy Efficiency (ASEE) to measure the energy efficiency of a specific cloud application. ASEE equals to the workload of the application divided
by the energy consumed by the application. The authors evaluated energy efficiency of three cloud applications which are a transactional Web e-commerce benchmark application, a CPU intensive application and a data intensive application, respectively. The authors claimed that the more resources used by a single application, the better its ASEE and its performance.

Although these approaches take into account the cloud application workload and system configurations when profiling and analysing the energy efficiency of the cloud system, they only focus on one particular workload type at one time. They do not consider different mixtures of applications running simultaneously. In real-world cloud environments, users send mixed type of workload to cloud systems simultaneously. The way different types of workload are composed and deployed will impact the energy efficiency of cloud application. Therefore, it is essential to investigate how different workload and resource allocation strategies impact energy efficiency in cloud systems.

2.5 Summary

In this chapter, the recent work related to performance and energy evaluation in cloud systems was reviewed. We identified three main related research areas: cloud computing, cloud system performance evaluation, and energy efficiency in cloud systems. We reviewed related work in each area and analysed its key limitations. We first identified the requirements of performance and energy consumption evaluation in cloud systems. Then we investigated the existing approaches in cloud system performance evaluations. At last, we analysed existing approaches for improving energy efficiency in cloud systems, including power and energy models, energy saving policies, and the energy profiling and analysis. The key research gaps in the abovementioned three research areas are summarised as following:

- **Energy Efficiency Evaluation Requirements.**

  High energy consumption in cloud systems has become a critical concern for both financial and environmental reasons. In addition, one of the important requirements for a cloud application is to provide reliable QoS. Ideally, the performance of a cloud system must not be jeopardised by the energy...
consumption minimisation. Therefore, a thorough understanding of the performance and energy consumption patterns of cloud applications is imperative.

- **Evaluating Performance.**
  Extensive research efforts have been made in evaluating performance of cloud applications. The key limitation we found in this area with regard to our research problem is that most of the approaches are simulation-based modelling which may have inaccurate evaluation results, as any incorrect environment settings have the potential to alter the results of the simulation and give the wrong results. Compared to simulation, performance test-bed can get more accurate performance test results in real-world cloud environments. However, there are no unified supporting tools to generate cloud workloads, collect and analyse both performance and energy data in an automatic manner.

- **Improving Energy Efficiency.**
  Research efforts in this area are to designing power and energy consumption models to predict cloud system energy consumption, applying energy saving techniques to dynamically change system management strategies and configurations, and profiling and analysing energy consumption to obtain guidelines for cloud systems design and maintenance. The existing power and energy consumption models focus mainly on single hardware components of cloud servers, such as CPU, memory, disk, etc. None of the existing power or energy consumption models can characterise the energy consumption of cloud applications. Moreover, most research efforts in the energy saving policies can only be applied in existing cloud systems. However, it is critical to perform what-if energy efficiency analysis before real deployment. Furthermore, a key limitation of existing energy profiling and analysing approaches is that they mainly focus on one particular workload type at one time. It is essential to investigate how different workload and their composition impact energy efficiency in cloud systems.
Chapter 3

Approach

In Chapter 1, we stated the key research challenges and problems related to system performance and energy consumption of cloud applications. We have reviewed the current research efforts and their limitations in Chapter 2. Based on this analysis, in this chapter we give an overview of our research approach to the analysis into the trade-off between system performance and energy consumption of cloud applications.

Section 3.1 briefly presents the research problems and overviews our approach. Section 3.2 introduces the high-level process of our approaches and the detailed solutions to each of the key research issues, including the energy consumption model (Section 3.2.1), the analysis and profiling of the performance and energy consumption of cloud applications (Section 3.2.2), the energy consumption signatures (Section 3.2.3) and the performance and energy consumption analysis tool (Section 3.2.4).

3.1 Overview

As discussed in Chapter 1, one of the most important requirements for a cloud application is to provide reliable QoS to end-users. Ideally, the performance of a cloud application must not be jeopardised by the process of minimising its energy consumption. Before the deployment of a cloud application, the characteristics of the system performance and energy consumption with various application deployment strategies and system configurations needs to be analysed, including the application structure and deployment. A thorough understanding of the performance and energy consumption of the application in a cloud environment is imperative. We need to learn how energy consumption and cloud system performance are affected by different workloads, application configurations and deployment platforms.
respectively. However, there might be hundreds even thousands of different deployment strategies for one single cloud application. Finding out which one is the most energy efficient is a significant task, as it requires:

- setting up the target cloud deployment environment;
- configuring and deploying the cloud application;
- running extensive experiments with different parameters/metrics and workloads;
- collecting appropriate and diverse cloud application energy/performance measurements during these experiments;
- performing an energy/performance trade-off analysis for each experiment’s results; and
- comparing results across experiments.

To this end, a solution that addresses the abovementioned tasks is needed. Such a solution must be able to efficiently determine the best energy-efficient cloud application deployment strategy based on collected performance and energy consumption data. In this thesis, we provide such a solution by:

- developing an energy consumption model which can characterise the energy consumption of cloud applications and can be applied as principles for collecting appropriate energy consumption data;
- conducting extensive experiments to validate our energy consumption model and analyse the trade-off between performance and energy consumption based on collected data;
- extracting energy consumption patterns of cloud applications from the experimental results; and
- developing a tool to support automatic load test and data collection for trade-off analysis between performance and energy consumption.

In Section 3.2, we briefly introduce the general approach and methods that define our solution.
3.2 Analysing Performance and Energy Consumption of Cloud Applications

Figure 3.1 shows the high-level process of developing our performance and energy consumption trade-off analysis solution. Four main research problems need to be addressed in order to deliver this solution. The detailed approach to these research problems are discussed in Section 3.2.1, Section 3.2.2, Section 3.2.3 and Section 3.2.4 respectively.

![Figure 3.1 High-level Solution Development Process](image)

3.2.1 Energy Consumption Model

As discussed in Chapter 2, the key limitation of much existing research on cloud energy consumption models is that they are predominantly focused on the processing resources, i.e. computation servers, since processing resources contribute a major part of the total energy consumption [85]. However, earlier research results indicate that at least 30% of the total energy is consumed by communication links, switching and aggregation resources [55]. In addition, data retrieval from storage resources also largely contributes to energy consumption when the size of the transferred data grows significantly and the number of fetch/store operations is large [86]. Few energy consumption models have taken these network and storage resources into account.

To overcome this limitation, we propose a new energy consumption model for cloud applications. In our energy consumption model, we consider storage, computation and communication resources as the energy consumed by those
resources are all important and highly dynamic in cloud environments. Although the energy consumed by the cooling system in cloud data centre environments can also be large [87], the cooling overhead can be approximately modelled as a fixed component of the total energy consumption [62]. Similarly, the energy consumption that occurs during idle time is also a fixed part of the total energy consumption in cloud environments. The dynamic part of the energy consumption is the part consumed by the running applications in cloud environments.

We thus divide the energy consumption into two parts: fixed energy consumption (energy consumed during idle time) and variable energy consumption (additional energy consumed by applications). Our energy consumption model is focused on the variable energy consumption and the fixed part is used as benchmark. In addition, the energy consumption of cloud applications vary greatly with different types of cloud applications and their workloads, as well as system configurations [14]. For instance, a video processing cloud application mainly consumes CPU cores, an on-line transaction processing application predominantly data querying and storage, and a social networking cloud application mainly consumes network resources such as communication message transmission. Different cloud applications consume different resources so that the energy consumption will differ. These factors, which can impact the energy consumption of cloud applications, have also been taken into account in our energy consumption model, the details of which are presented in Chapter 4.

3.2.2 Performance and Energy Consumption Trade-Off Analysis

Before being applied in the energy consumption data collection process, our energy consumption model needs to be validated. Therefore, extensive experiments have been conducted to profile and analyse energy consumption of cloud applications. Meanwhile, system performance has also been analysed as we were aiming to identify the relationship between performance and energy consumption.

As discussed in Section 3.2.1, the types and workloads of cloud applications, as well as system configurations, are considered in our energy consumption model. We focused on the variety of these factors when we collected and analysed system
performance and energy consumption data. Our experiments can be categorised into the following three major groups:

1. **Changing the types of cloud applications.** The energy consumption of different resources in cloud environments vary significantly. Different cloud applications can impose substantially different resource requirements. Even the resource requirements of a particular application can vary over time. Thus, we investigated how the types of cloud applications can influence the system performance and energy consumption. In cloud environments, users send mixed types of requests to different cloud applications simultaneously. The way those different types of applications are composed and deployed will impact the performance and energy consumptions of the cloud applications. Therefore, we also profiled and analysed the performance and energy consumption of mixed types of cloud applications in our experiments.

2. **Changing workloads.** For a single cloud application, different numbers of user requests and different content in those requests result in different execution time of the cloud application. Accordingly, the system performance and energy consumption will differ. Hence, we studied how the workload of a single cloud application can impact system performance and energy consumption. Furthermore, the total energy consumption of two co-located cloud applications is not equivalent to the sum of the energy consumed individually. Thus, we also analysed the interference caused by resource contention when multiple cloud applications were running on the same server/VM simultaneously.

3. **Changing system configurations.** The system performance and energy consumption of the same application executing on different VMs can also vary significantly. Thus, we changed the configurations of the VMs allocated to the cloud application in the experiments, including the size of each VM and the service deployment strategies we applied to deploy the application.

The results of the performance and energy consumption profiling and analysis are presented in Chapter 5.
3.2.3 Energy Consumption Signature

Due to the variety of cloud application workloads and system configurations, it is time consuming to perform load test and collect data in all the different scenarios. Therefore, it is necessary to provide guidelines to cloud system architects and performance engineers. These guidelines will reduce their effort in investigating system performance and energy consumption under specific system configurations and workloads. Thus, a series of generic energy consumption patterns need to be identified, which should be able to summarise the system performance and energy consumption of cloud applications.

In order to achieve the abovementioned objective, extensive experiments have been conducted to analyse the system performance and energy consumption as discussed in Section 3.2.2. Based on the results of our analysis, a series of energy consumption patterns was derived and codified. These patterns are formalised as “signatures”. Each signature specifies a set of invariants that indicate the likely system performance and energy consumption of the cloud application under a given system configuration and workload. Therefore, by matching the system configuration and workload of the scenario under test to the system configurations and workloads specified in the signatures, cloud system architects and performance engineers can predict the system performance and energy consumption without running load test. We adopt the declarative and formal Object Constraint Language (OCL) [88], a well-known and extensible language to capture such signatures.

To support formalise these energy consumption patterns, we have developed a cloud system architecture model and cloud application workload model:

- **Cloud System Architecture Model**: Different cloud system configurations and resources allocation strategies have been applied in the experiments as introduced in Section 3.2.2. In order to capture the characteristics of a cloud system, an abstract cloud system architecture model has been designed. This abstract cloud system architecture model includes all necessary infrastructure elements such as physical servers and VMs in cloud environments, the attributes of each element and the relationship between those elements.
• **Cloud Application Workload Model**: Various cloud application workloads have been applied during the performance and energy consumption evaluation experiments. A workload model is needed to capture both cloud users’ behaviours and cloud application data. A form chart model is a technology-independent bipartite state diagram used to simulate user behaviours of submit/response systems [89]. It describes what the user sees as system output and what the user provides as input to the system at a high level. It captures the structure of the target system from users’ perspectives and can be augmented with probabilities to capture user interactions with the target system. A *stochastic form chart model* has been extended from the basic form chart model with stochastic functions to generate performance testing workloads of Web applications [90, 91]. In the stochastic form chart model, the pages of a web site are represented as bubbles, the actions as boxes and the transitions between them as arrows. In addition to the form chart, a form-oriented model specifies the message types and user data for all the pages and actions. The stochastic form chart can be extended to model the cloud system workloads. Cloud users send service requests to the cloud system and receive responses correspondingly. Therefore, a cloud system can be considered as a submit/respond system. User behaviours of requesting cloud services can be modelled using stochastic form charts. Thus, we adopted the *stochastic form chart* to model realistic cloud application workloads.

The details of energy consumption signatures are presented in Chapter 6.

3.2.4 **Performance and Energy Consumption Analysis Tool for Cloud Applications**

Profiling and analysing system performance and energy consumption of cloud application is time consuming. Extensive experiments with different parameters, metrics and workloads need to be conducted. Manual generation of load test plans, change of system configurations and application of load test are very tedious and error-prone. In addition, most existing approaches to cloud system performance and
energy consumption profiling limit the types of tasks running in the profiling process to only discrete individual types [15]. One major reason for this limitation is the difficulties in manual application of different workloads to multiple cloud applications. In order to overcome this limitation, we have developed *StressCloud*, an automatic performance and energy consumption analysis tool for cloud applications.

The StressCloud performance and energy consumption analysis tool is expected to support alternative architectural choices for performance and energy evaluation of cloud applications. Thus, different cloud system configurations will be applied in the evaluation process. In order to achieve this goal, we integrated the cloud architecture model introduced in Section 3.2.3 into StressCloud. Therefore, system architects and performance engineers can change system configurations by adding/removing the elements and modifying the attributes of the elements.

Various cloud system workloads need to be applied during the performance and energy consumption evaluation process based on different workload models. In order to conduct a reliable system performance and energy consumption evaluation, a realistic and comprehensive cloud system workload model is required. Thus, we integrated the cloud application workload model introduced in Section 3.2.3 into StressCloud.

*Figure 3.2 Overview of Performance and Energy Consumption Trade-off Analysis Tool*
3.2.4.1 High-level Tool Process

Figure 3.2 shows the overview of our performance and energy consumption analysis tool. The cloud architecture model (1) is defined by the end user of StressCloud. The structure of deployed application and corresponding workload models can be abstracted and generated automatically on both application structure and application data (2) specified by the end user. The deployment scripts will then be automatically generated by the deployment script generation engine (3). The generated workload model will be translated into load test scripts (4). Based on the system deployment and workload test scripts, load test will be performed automatically (6). Based on the energy consumption model (6), the energy consumption data (6b) and performance measurements (6a) will be automatically collected during the load test process. Then, the end user will analyse the trade-off between system performance and energy consumption of the cloud application (9). The suggested energy-efficient deployment strategy can be added to the energy consumption signature data set (7) for future reference (10). Before performing the load test, the end user can validate the architectural model and workload model in the energy consumption signature data set (7). The predicted system performance and energy consumption will be presented to the end user (8). Thus, the end user can choose to change the model or run the load test to get real data. The model design and tool implementation are presented in Chapter 7 in details.

3.2.4.2 Evaluation

In order to evaluate StressCloud, we used it to conduct a number of cloud application energy and performance analysis case studies. We profiled and analysed system performance and energy consumption of three different cloud applications, named JPetStore\textsuperscript{7}, Pebble\textsuperscript{8} and ImageProcessor\textsuperscript{9}, respectively. We applied different workloads in the load test for each cloud application. Each workload focused on different aspects of the cloud application’s user behaviours. For instance, as JPetStore is an e-commerce cloud application, we analysed the performance and energy consumption of JPetStore with workloads composed of “Search” and

\textsuperscript{7} http://java.sun.com/developer/releases/petstore/
\textsuperscript{8} http://pebble.sourceforge.net/
\textsuperscript{9} http://imagej.nih.gov/ij/
“Order” requests, respectively. We also changed the application deployment strategies under certain workloads to investigate the energy efficiency of each deployment decision. The details of these case studies are presented in Chapter 8.

3.3 Summary

In this chapter, we outlined our solution to performance and energy consumption trade-off analysis of cloud applications. Specifically, we discussed in detail our approach to the following research issues: 1) the energy consumption model for cloud systems; 2) the performance and energy consumption analysis of cloud applications; 3) the energy consumption signatures; and 4) the performance and energy consumption analysis tool. The following chapters explain the research work conducted into each of these areas in detail.
Chapter 4

Energy Consumption Model

Prior to the analysis of the trade-off between system performance and energy consumption, an energy consumption model to analyse energy consumption data of cloud applications is essential. This chapter presents our proposed novel energy consumption model for cloud applications. It provides a basis for characterising energy consumption of cloud applications under different workloads and system configurations.

This chapter continues as follows. Section 4.1 gives an overview of current approaches’ limitations. Section 4.2 introduces the energy consumption model proposed, including the model structure, energy calculation unit and energy calculation formula. Section 4.3 demonstrates the usage of our energy consumption model. Section 4.4 discusses the key advantage and the key limitation of our energy consumption model. Section 4.5 summarises this chapter.

4.1 Introduction

The major aim of approaches for energy saving and optimisation in cloud computing is to create the cloud system using low-energy components and keep the performance of cloud system at an acceptable level. In most cloud systems, CPU usage contributes to 35%-50% of a cloud server’s total power consumption [92, 93], which makes them the most energy-consuming component of the cloud system. In addition, the storage devices can drain a significant amount of power of a cloud server, even when idle (spinning but not performing an operation) [94]. For instance, a server class IBM Ultrastar 36ZX [95] disk is rated at 22.3 Watts (comparing this to an Intel Xeon processor clocked at 1.6 GHz which is rated at 57.8 Watts). The other main energy-consuming component is the memory which consume about 27% of a cloud server’s total power consumption [96]. Thus, most of existing research on
energy efficiency of cloud systems have primarily focused on the optimisation of the energy consumption of processing elements such as computation servers. However, computation servers contribute only partially to the energy consumed by cloud data centres. About one-third of the total energy consumption of a cloud data centre are caused by communication links, switching, and aggregation elements [97]. In addition, data retrieval from storage resources also contributes a significant part of total energy consumption, with data size growing significantly [86]. Moreover, energy consumption of cloud data centres may vary greatly with different system resource configurations and allocation strategies, as well as the workload in cloud environments [14]. However, none of the existing cloud energy consumption models have taken them into account. Therefore, we propose a new energy consumption model for analysing energy consumption of cloud applications. The details of our energy consumption model are presented in Section 4.2.

4.2 Modelling Energy Consumption of Cloud Applications

In this section, we first introduce the setting of our energy consumption model in Section 4.2.1. Then we discuss the energy consumption calculation unit in Section 4.2.2. Finally, we present the usage of our energy consumption model using an example cloud application in Section 4.2.3.

4.2.1 Model Setting

As introduced in Section 4.1, the major energy consumption units in cloud environments are cloud servers. The rest of the energy consumption is caused by the cooling system. Although it is usually high in cloud computing infrastructure environments [87], the cooling overhead can be approximately modelled as a fixed component of the total energy consumption [62]. Therefore, we focus on storage, computation and communication resources as the energy consumed by them are both significant and highly dynamic in cloud environments.

Energy consumed during idle time is also a fixed part of the total energy consumption. According to [98], a server with zero workload consumes about 60% of its peak power. We monitored the power consumption of a cloud server in our
private cloud environment - SwinCloud. The model of the cloud server is HPZ400 which has four physical cores. We noticed that the power consumption of the server in idle state was 90 watts on average. Then we ran a cloud application which calculates Fibonacci sequence based on a given number. The power consumption of the server was close to 180 watts on average when the CPU usage of the server reached 100%. Therefore, the dynamic part of energy consumption is the additional energy consumed by running applications on cloud environments. Thus we further divide the energy consumption into two parts: the fixed part (energy consumed during idle time) and the variable part (additional energy consumed by cloud applications). In this thesis, we focus on the variable part of energy consumption in cloud environments.

4.2.2 Energy Calculation Unit

More and more customers have adopted the cloud to fulfil their IT infrastructure needs. Cloud service providers need to be prepared for handling highly heterogeneous workloads. Hence, analysing the impact that specific workloads have on energy consumption is critical. As defined in [99], workload is “the amount of work assigned to, or done by, a client, workgroup, server, or system in a given time period”. In the context of cloud computing, workloads are the different tasks submitted by all the customers and executed in the cloud data centres.

The workload of a cloud application is a set of tasks that submitted to the cloud and processed by the application. A cloud application consists of several functional service units. Each service unit utilises some mix of computation, storage and communication resources. However, the percentage of each resource used by different service units is significantly different. Therefore, service units can be classified as computation-intensive, data-intensive or communication-intensive based on the major resources consumed in the cloud environment [100]. A computation-intensive service unit mainly consumes CPUs or memories. A data-intensive service unit mainly processes I/O bound service requests. A communication-intensive service unit transfers large amount of data among cloud servers. Every service unit can be considered as a queue that holds incoming tasks. Based on the type of service unit that a task requires, we can categorise the tasks of a
cloud application into three types: computation-intensive, data-intensive and communication-intensive. Thus, the workload of a cloud application can be represented as either a set of single type of tasks or a composition of the abovementioned two or three types of tasks.

Instead of measuring the energy consumption of individual hardware components, we consider a single task running in cloud environments as the fundamental unit for energy profiling. A task can be defined as a three-tuple denoted as \((input, process, output)\). For instance, a task to zip a file can be denoted as (source file, zip, target file). If the input can be partitioned and processed separately, we consider each partition as an individual task. For example, a 10GB file may be processed as a single file or divided into two 5GB files to be processed by separate tasks. Tasks themselves have properties or attributes specified in “input” that describes their behaviours. These attributes are usually expressed as the type and amount of resources the task consumes, the specific system configuration and any QoS constraints.

In order to demonstrate the impact of energy consumption caused by different types of tasks, we present the details of each type of task as follows:

1. For a computation-intensive task \(t_{\text{comp}}^i\), \(1 \leq i \leq n\), where \(n\) is the total number of computation-intensive tasks, the energy consumption of \(t_{\text{comp}}^i\) is defined as \(EC(t_{\text{comp}}^i)\).
2. For a data-intensive task \(t_{\text{data}}^i\), \(1 \leq i \leq m\), where \(m\) is the total number of data-intensive tasks, the energy consumption of \(t_{\text{data}}^i\) is defined as \(EC(t_{\text{data}}^i)\).
3. For a communication-intensive task \(t_{\text{comm}}^i\), \(1 \leq i \leq k\), where \(k\) is the total number of communication-intensive tasks, the energy consumption of \(t_{\text{comm}}^i\) is defined as \(EC(t_{\text{comm}}^i)\).

For example, a Web service which calculates a Fibonacci sequence based on the largest number of the sequence is deployed on one SwinCloud server. The largest number – defined as \(LN\) - is specified by the cloud user. This service mainly consumes CPUs and is running on a VM which has four virtual CPUs and 8GB RAM. If the cloud user submits one task requesting calculating Fibonacci sequence
with $LN = 52$, it takes 66 seconds to complete the calculation on average. In this case, the total number of computation-intensive task equals to one. The difference of average power consumption between the server with and without workload is 88 watts. Therefore, the energy consumption of the computation-intensive task $EC(t_{\text{comp}}^i)$ is 5860 Joule which equals to 88 watts multiplied by the task execution time 66.6 seconds. For data-intensive tasks and communication-intensive tasks, the explanations of the terms are similar to computation-intensive tasks.

4.2.3 Energy Calculation Formula

As discussed in Section 4.2.1, the total energy consumed in cloud environments is composed of fixed energy consumption, defined as $EC_{\text{Fix}}$, and variable energy consumption, defined as $EC_{\text{Var}}$. Thus, the total energy consumption, noted as $EC_{\text{Total}}$, is formulated as:

$$EC_{\text{Total}} = EC_{\text{Fix}} + EC_{\text{Var}} \quad (4.1)$$

The total energy consumed in cloud environments is the sum of energy consumed by all tasks. Different types of tasks simultaneously exist in cloud environments. In our model, the total energy consumption is defined as:

$$EC_{\text{Total}} = EC_{\text{Fix}} + \sum_{i=1}^{m} EC(t_{\text{comp}}^i) + \sum_{i=1}^{m} EC(t_{\text{data}}^i) + \sum_{i=1}^{l} EC(t_{\text{comm}}^i) \quad (4.2)$$

For each task, the energy consumption is tightly coupled with task parameters. For instance, for a computation-intensive task, the energy consumption increases with the number of processes used by the task [14]. There are other specific application-related factors that can influence the energy consumed by a task. For instance, the encoding and decoding algorithms of data-intensive tasks, e.g. video streaming, may impact the complete time of the data-intensive task. Therefore, the energy consumption of the data-intensive task may be different with different encoding and decoding algorithms. However, we focus on the factors which are related to a task workload. Those factors, shared by most tasks, directly and largely influence energy consumption. For example, the size of raw data processed by a task of movie-making application. In addition, system configurations have significant
impact on energy consumption. Energy consumption increases dramatically when the number of VMs configured on a physical machine increases [101].

In our model, the parameters of a task taken into account in the calculation of energy consumption include: the number of processes for the task, defined as $PT$; the size of data to be processed, defined as $DT$; and the type of operation it requires, defined as $OT$. The system configurations, such as hardware of the physical server and the scale of the configured VMs, also have significant impact on energy consumption. Hence, the energy consumed by each task is determined by the task parameters and the system configuration. We denote the system configuration as $C$.

Thus, the energy consumption of each type of task is defined as:

$$EC(t_{\text{comp}}^i) = f_{\text{comp}}(PT_{\text{comp}}^i, DT_{\text{comp}}^i, OT_{\text{comp}}^i, C_{\text{comp}})$$  \hspace{1cm} (4.3)

$$EC(t_{\text{data}}^i) = f_{\text{data}}(PT_{\text{data}}^i, DT_{\text{data}}^i, OT_{\text{data}}^i, C_{\text{data}})$$  \hspace{1cm} (4.4)

$$EC(t_{\text{comm}}^i) = f_{\text{comm}}(PT_{\text{comm}}^i, DT_{\text{comm}}^i, OT_{\text{comm}}^i, C_{\text{comm}})$$  \hspace{1cm} (4.5)

### 4.3 Example Usage

We conducted a set of primary experiments to investigate the energy consumption of cloud applications with different workloads and system configurations. In this Section, we present the details of the experiments to demonstrate the usage of the energy consumption model.

The experiments were run on a single physical server in SwinCloud. The server deployed in SwinCloud is HPZ400. Each server has four physical cores and 10GB RAM. The frequency of the CPU is 2.8 GHz. The Java Web application JPetStore was deployed on the cloud server. JPetStore uses a Web server to handle the user requests and a database server to process the database queries in response to the user requests. Therefore, its workload is composed of communication-intensive task, data-intensive task and a small proportion of computation-intensive task. For our experiment, we used its communication aspects only as the major proportion of its workload is communication-intensive task. We arranged the workload to minimize
the load on sender/receiver in order to measure the energy consumption of the communication aspects only of this reference application.

All network traffic generated by customers was sent from a client PC and emulated using JMeter\(^{10}\) which is a load test tool for analysing and measuring performance of the variety of Web applications. Each user session generated by JMeter consisted of a series of sequential user requests, such as browsing the list of products, adding items to the shopping cart, checking out, and so on. Each run of JMeter was considered as a communication-intensive task \(t_{\text{com}}^i\) \((1 \leq i \leq k)\) where \(k\) was the total number of runs. For each communication-intensive task, the parameters included the user request type \(OT_{\text{com}}^k\), the number of requests in each session \(PT_{\text{com}}^k\) and the packet size of each user requests \(DT_{\text{com}}^k\).

We first profiled the power consumption of the cloud server in idle state. Then we profiled the power consumption of the cloud server with different workloads and system configurations \(C_{\text{com}}^k\). The energy consumption of JPetStore equalled to the difference of average power consumption between the server with and without runtime tasks multiplied by the execution time of the tasks.

Two major sets of experiments were designed and conducted to demonstrate the usage of the energy consumption model. As presented in Section 4.2.3, system configuration and parameters of tasks have been identified as the two major types of factors which can impact energy consumption of cloud applications. Thus, in each set of test, we first changed the system configuration \(C_{\text{com}}^k\) to demonstrate its impact on energy consumption. For each set of test, we first deployed the Web server on a VM which had two virtual CPUs and 4GB RAM, and the database server on a VM which had one virtual CPU and 2GB RAM. We use \(C^1\) to represent this configuration. Then we changed the system configuration \(C_{\text{com}}^k\) by changing the Web server from a VM which had two virtual CPUs and 4GB RAM to a VM which had one virtual CPU and 2GB RAM. We use \(C^2\) to represent this configuration. In addition, we also changed the task parameters of communication-intensive tasks to

\(^{10}\) http://jmeter.apache.org/
demonstrate their impact on energy consumption. The details of the experiments are presented as following:

1. **Test set # 1: keep the packet size of each user request** $DT^k_{\text{comm}}$ **constant,**
   while changing system configuration $C^k_{\text{comm}},$ user request type $OT^k_{\text{comm}},$
   and **the number of requests in each session** $PT^k_{\text{comm}}.$

During testing, we changed the total number of user sessions $PT^k_{\text{comm}}$ and the request types in each user session $OT^k_{\text{comm}}.$ Each user session consisted of mixed browsing requests or shopping requests in different percentages. There were five types of mixed user requests, as presented in Table 4.1. Browsing requests included checking home page, viewing catalogue, viewing products, searching products and so on. Shopping requests included checking out, updating shopping cart, filling order forms, ordering inquiry, and so on. For each type of mixed user requests, we increased the number of concurrent user sessions from 100 to 700 in steps of 100. The energy consumption of the application with system configuration $C^1$ and $C^2$ are presented in Figure 4.1 and Figure 4.2. As the number of concurrent user sessions $PT^k_{\text{comm}}$ increased, the energy consumption increased when we fixed the system configuration $C^k_{\text{comm}}.$ This was due to the increase in the number of concurrent user sessions $PT^k_{\text{comm}}$ causing extra scheduling overhead as each user request was processed. In addition, we noticed that Mixed 1 consumed the most energy while Mixed 5 consumed the least energy among all five workload types, despite the type of VM. Since Mixed 1 contained the most browsing requests and Mixed 5 contained the most shopping requests, Mixed 1 had the most disk and memory access of all as opposed to Mixed 5. Thus, it took longer to complete requests of type Mixed 1 compared to the other four workload types. As a result, the energy consumption of Mixed 1 was more than the other four workload types.

**Table 4.1 Mixed Type of User Requests**

<table>
<thead>
<tr>
<th></th>
<th>Mixed 1</th>
<th>Mixed 2</th>
<th>Mixed 3</th>
<th>Mixed 4</th>
<th>Mixed 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsing Request</td>
<td>100%</td>
<td>80%</td>
<td>50%</td>
<td>20%</td>
<td>0%</td>
</tr>
<tr>
<td>Shopping Request</td>
<td>0%</td>
<td>20%</td>
<td>50%</td>
<td>80%</td>
<td>100%</td>
</tr>
</tbody>
</table>
2. Test set # 2: keep user request type $\mathcal{OT}_{\text{comm}}^k$ and the number of requests in each session $\mathcal{PT}_{\text{comm}}^k$ constant, while changing system configuration $\mathcal{C}_{\text{comm}}^k$ and the packet size of each user request $\mathcal{DT}_{\text{comm}}^k$.

We fixed the number of concurrent user sessions $\mathcal{PT}_{\text{comm}}^k$ at 300. First, we ran the test with only browsing requests and gradually increased the packet size $\mathcal{DT}_{\text{comm}}^k$ of the browsing requests on different VM of different types. The packet size of each browsing request is listed in Table 4.2. These packet sizes represent the majority of all the browsing requests of JPetstore. The packet size of request “Browsing 1” is the smallest while packet size of request “Browsing 3” is the largest. The corresponding system energy consumption is presented in Figure 4.3. As demonstrated, there was a slight increase in the energy consumption of each task when we increased the packet size of the requests. Bigger packet size $\mathcal{DT}_{\text{comm}}^k$ usually leads to more transmission...
time over the network and more processing time on both the servers and switches. Accordingly, the processing time of each user requests is longer and the energy consumption increases for each communication-intensive task. In addition, the energy consumption of each task under system configuration \( C^2 \) was more than the task energy consumption under system configuration \( C^1 \), no matter what the packet size was. We observed that when the size of the VM deployed for the Web server decreased (from \( C^i \) to \( C^j \)), the energy consumption increased in general. Intuitively, the more resources used by the VM the greater the energy consumption. However in this case, the smaller the instance the higher the disk accesses due to the thrashing of the cache, which leads to increase in energy consumption. Even for the VM which has two cores, the extra energy consumption by the additional core was cancelled out by the much larger added memory, reducing the number of accesses to the database server.

**Table 4.2 Browsing Request Packet Size**

<table>
<thead>
<tr>
<th>Packet Size(KB)</th>
<th>Browsing 1</th>
<th>Browsing 2</th>
<th>Browsing 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3758</td>
<td>4190</td>
<td>4853</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.3 Energy Consumption of JPetStore with Increased Packet Size and Different System Configurations**

### 4.4 Discussion

We have proposed an energy consumption model for profiling and analysing energy consumption of cloud applications. In this energy consumption model, we treat a single task as a unit of calculating the energy consumption of cloud applications. Based on the type of service unit in the cloud application that a task requires, we
categorise the tasks of a cloud application into three types: computation-intensive, data-intensive and communication-intensive. For each type of task, we identified that the energy consumption of a task is related to system configurations and parameters of the task, i.e. the number of processes for the task $PT$, the size of data to be processed $DT$, and the type of operation it requires $OT$.

As presented in Section 4.3, we have demonstrated how our energy consumption model can be applied to analyse the energy consumption of JPetStore with communication-intensive workload. The preliminary experiment results show that the energy consumption of JPetStore is highly coupled with system configurations and application workloads. More experiments have been conducted to further validate the energy consumption model and analyse the energy efficiency of various cloud applications with different system configurations and workloads. The details of the experiment results will be introduced in Chapter 5. Based on the validation results of the energy consumption model, a number of cloud applications’ energy consumption patterns of different system configurations and workloads have been extracted. The details of these patterns will be presented in Chapter 6. These patterns will be used for prediction analysis of a cloud application with given system configurations and workloads.

The major limitation of our energy consumption model is that it does not give the coefficients’ value of the variants, i.e. system configurations and task parameters, in the formulas. Therefore, it cannot be used to calculate the specific amount of energy consumption for a given cloud application. The reason behind this is the heterogeneity of cloud system configurations and cloud application workloads. Huge number of experiments with different system configurations and cloud application workloads need to be conducted in order to find out the shape of the functions even if there is only one variant. However, our major research objective is to help system architects or performance engineers to determine best energy-efficient cloud application deployment strategy based on collected performance and energy consumption data, as discussed in Chapter 3. For a given cloud application, the best energy-efficient deployment strategy can be determined if the trade-offs between performance and energy consumption has been found. Our
energy consumption model can be applied to analyse or predict the trend of cloud applications’ energy consumption when the values of the variants change.

### 4.5 Summary

In this chapter, we presented our new energy consumption model for cloud applications. In this energy consumption model, we considered a single task running in cloud environments as the fundamental unit for energy profiling. All runtime tasks in cloud environments were categorised into three types: computation-intensive, data-intensive and communication-intensive. To validate this model, we conducted extensive experiments to profile the energy consumption of cloud applications in cloud environments based on different types of the tasks. We collected fine-grained system performance and energy consumption data with varying workloads and system configurations. The details of the experiments will be presented in Chapter 5.
Chapter 5

Performance and Energy Consumption Profiling and Analysis

To validate the energy consumption model presented in Chapter 4, we have conducted extensive experiments to profile and analyse system performance and energy consumption by running heterogeneous workloads in a real-world cloud environment. Based on the energy consumption profiling results, the impacts of the factors indicated in the energy consumption model on the energy consumption of different tasks are verified. Our objective was to collect data and analyse the relationship between energy consumption, system configuration, workload, and system performance. The analysis results were also adopted to derive the energy consumption patterns introduced in Chapter 6. In this chapter, we discuss the details of the experiments, including experimental setup, test case design and results analysis.

This chapter continues as follows. Section 5.1 gives a brief introduction to the experiments. Section 5.2 discusses the experiments setup. The test case design of all experiments is presented in Section 5.3. We analyse the experiment results in Section 5.4. Conclusions drawn from the experiment results analysis are discussed in Section 5.5. Section 5.6 summarises this chapter.

5.1 Introduction

In our energy consumption model, runtime cloud tasks are divided into three types based on the type of service unit that a task requires: computation intensive, data intensive and communication intensive. We profiled the energy consumption by creating heterogeneous VMs in a real-world cloud environment and running the abovementioned three types of tasks. For each type of task, we also profiled the
energy consumption and system performance with various task parameters, that is, the number of processes $PT$, the size of data to be processed $DT$ and the type of operation it requires $OT$. We investigated the system performance and energy consumption with workloads composed of both single type of task and mixed types of tasks.

The experimental setup, test case design and experiment results are presented in the following sections.

5.2 Experimental Setup

In this section, we first present the test-bed used to profile the energy consumption data in Section 5.2.1. Then we discuss the profiling method in Section 5.2.2.

5.2.1 Test-bed

SwinCloud is a private cloud that provides a common computational infrastructure to researchers at Swinburne University of Technology. Our experiments were conducted in SwinCloud as we have detailed knowledge of the cloud hardware, networking, operating system versions, and other application software. SwinCloud was deployed in the Energy Research Lab (ERL) at Swinburne University of Technology. This lab focuses on energy-related research and development. Using the extensive and sensitive power monitoring facilities provided by this lab, we could precisely monitor the power consumption of the SwinCloud server and its network devices, including network cards, switches and routers. The power consumption measurement was realised and managed using PowerNode, a power usage profiling equipment developed by GreenWave Reality\footnote{http://www.greenwavereality.com/}. It supports measurement of both immediate and average power consumption. Collected power readings were reported to the GreenWave Gateway, which is used for creating a mesh-based Home Area Network (HAN). The GreenWave Gateway then sent the data to the GreenWave Reality data centre, where the information could be accessed and analysed.
Multiple servers exist in a cloud environment. The energy consumption in the cloud environment includes the energy consumed by individual servers and the scheduling and communication overhead across different servers. In this thesis, we focus on the energy consumption of individual servers as it is the predominant part [102]. Furthermore, the cross-server scheduling and communication overhead in one cloud environment can be significantly different from the other, depending on the scheduling mechanism adopted by the cloud and the distribution of the constituent servers. Thus, we conducted our experiments by measuring energy consumption of tasks running on a single isolated server. We left research to determine the energy consumption incurred by cross-server scheduling for future work. Moreover, we did not adopt any existing energy-saving policies in our experiments such as DVFS [11] or VirtualPower [103], in order to be able to isolate the factors causing unexpected energy consumption. Our experiment setup can however be reused to include these options.

<table>
<thead>
<tr>
<th>Basic Specification</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Cores</td>
<td>4</td>
</tr>
<tr>
<td>Number of Threads</td>
<td>2 Intel Hyper-Threading Technology</td>
</tr>
<tr>
<td>CPU Frequency</td>
<td>2.8GHz Fixed CPU Frequency</td>
</tr>
<tr>
<td>Memory</td>
<td>10GB Memory Speed 1333 MHz</td>
</tr>
<tr>
<td>Hard Disk Drive</td>
<td>1TB 7200 RTM SATA</td>
</tr>
<tr>
<td>Network Interface</td>
<td>Intel e1000 Gb</td>
</tr>
</tbody>
</table>

The type of the servers currently deployed in SwinCloud is HP Z400. Table 5.1 lists the specifications of the HP Z400. The Virtual Machine Manager (VMM) utilised for VM management is VMware\textsuperscript{12} ESX 4.1 and the operating systems running on the VMs are either Window Server 2008 or Windows 7 Professional. In the experiments, all VMs were assigned with 2GB, 4GB, 6GB or 8GB of RAM. The number of virtual CPUs (vCPUs) of each VM varied from 1 to 4 in steps of 1. The number of vCPUs equalled to the number of physical cores assigned to the VM. The configuration scales of the VMs are shown in Table 5.2.

\textsuperscript{12}http://www.vmware.com/
Table 5.2 Type of VM

<table>
<thead>
<tr>
<th>Virtual Machine</th>
<th>Number of Cores</th>
<th>RAM</th>
<th>Hard Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>2GB</td>
<td>80GB</td>
</tr>
<tr>
<td>Medium</td>
<td>2</td>
<td>4GB</td>
<td>80GB</td>
</tr>
<tr>
<td>Large</td>
<td>3</td>
<td>6GB</td>
<td>80GB</td>
</tr>
<tr>
<td>XLarge</td>
<td>4</td>
<td>8GB</td>
<td>80GB</td>
</tr>
</tbody>
</table>

Cloud Server

Figure 5.1 Performance and Energy Data Profiling Framework

Figure 5.1 presents the energy profiling framework for different tasks used in our experiments. In order to eliminate the impact of energy consumption and system performance introduced by energy collection applications, a PowerNode monitor was directly connected to the SwinCloud server. All workloads were generated and then sent to the SwinCloud servers from a client PC. We deployed a collection of Web services for load tests to the cloud server hosted on virtual machines. The details of the Web services deployed for each type of task are described in Section 5.3.
5.2.2 Profiling Method

The energy consumption of a task equals to the difference of average power consumption between the server with and without workload multiplied by the execution time of the task. We firstly retrieved the average power consumption measured by PowerNode with no hosted application workload in the cloud environment as our idle state benchmark. Then, we retrieved the average power consumption measured by PowerNode during the running time of each task. Finally, we multiplied the average power consumption by the execution time of the task to obtain the energy consumption of the task.

As discussed in Chapter 4, the task parameters, including the number of processes $PT$, the size of data to be processed $DT$, the type of operation it requires $OT$, system configuration $C$, have impact on the task energy consumption based on the energy consumption model. These task parameters determine the workload of a single task. In addition, the number of running tasks determines the overall workload of a cloud application. The system configuration $C$ includes two parts: 1) the amount and type of resources allocated to the service units of the cloud application; and 2) the specific parameters of each resource. Energy consumption is highly influenced by the workloads in the cloud environment as higher workloads consume more resources that lead to higher energy consumption. Moreover, system configurations, such as the number and scale of the configured VMs, also impact energy consumption significantly. In addition, workloads and system configurations may also impact system performance. As such, we took the workload and system configuration as inputs of our experiments. Energy consumption and system performance are outputs of our experiments.

We selected the throughput of the system as one of the key performance indicators (KPI). This is because throughput is often a key performance parameter monitored in cloud environments. It has the advantage of reflecting resource usage accurately [104]. The other KPI selected is the response time as it is a major performance QoS requirement in cloud environments [105]. For computation-intensive and communication-intensive tasks, the throughput is defined as the total number of user interactions requested and completed successfully per
second. There are two types of data-intensive task: file related which is focused on file processing and database related which is focused on database operations. For file related data-intensive tasks, the throughput is defined as the amount of data transferred per second. For database related data-intensive tasks, the throughput is defined as the total number of user interactions requested and completed successfully per second. The response time is defined as the interval from the initiation of a request to the receipt of the corresponding response.

5.3 Test Case Design

We designed and conducted five series of experiments. The first three series of experiments focused on individual types of tasks where we aimed to further investigate the impact of workload and cloud application deployment strategies on system performance and energy consumption of single type of tasks. The last two series of experiments focused on the mixed type of tasks, examining energy and performance for tasks with e.g. a 75% computation- and 25% data-intensive mix of workload. The objective of the last two series of tests was to apply the workloads of real-world cloud applications with more complexity and investigate the system performance and energy consumption with different cloud application deployment strategies, e.g. what happens when split data and compute load over different VMs instead of sending data and compute to the same VM? Only one aspect was changed in each test set in order to isolate the impact factors of system performance and energy consumption. The detailed test case design for the five series experiments are described in Sections 5.3.1, 5.3.2, 5.3.3, 5.3.4 and 5.3.5, respectively.

5.3.1 Computation-Intensive Task

The major resources consumed by computation-intensive tasks are the CPU cores and RAM. Thus, we can further divide the computation-intensive tasks into CPU-intensive tasks and memory-intensive tasks, respectively:

- **CPU-intensive tasks:** We deployed a cloud application containing only one CPU-intensive service unit which calculates a Fibonacci sequence as a representative CPU-intensive task. Each invocation of this service unit was a
CPU-intensive task $t_{\text{compt}}$. The largest number of the Fibonacci sequence determined the duration of each calculation – defined as $LN$. We mapped this number to the size of data to be processed of each CPU-intensive task. As introduced in Section 4.2.3, the energy consumption of a computation-intensive task $t_{\text{compt}}$ is related to the number of the processes for the task $PT_{\text{compt}}$, the size of data to be processed $DT_{\text{compt}}$, the type of operation it requires $OT_{\text{compt}}$, and the system configuration $C_{\text{compt}}$. In this case, the number of the processes for the task $PT_{\text{compt}}$ is the number of threads for each Fibonacci sequence calculation, the size of data to be processed $DT_{\text{compt}}$ is $LN$, and the type of operation of a CPU-intensive task $OT_{\text{compt}}$ is calculating a Fibonacci sequence. In order to analyse the impact of workload and system configurations $C_{\text{compt}}$ on the total energy consumption and throughput, we designed four sets of test cases in this series of experiments for CPU-intensive tasks. We first ran a single task to calculate the Fibonacci sequence and increased the $LN$ of the task gradually with fixed system configuration in test set 5.4.1.1.1 as described in Section 5.4.1.1. Then, we increased the resource allocated to the cloud application while keeping $LN$ constant. CPU-intensive tasks mainly require service units which consume CPU resources and the Hyper-Threading (HT) technology is usually applied to improve the overall performance of the CPU. Thus, we measured the energy consumption and system performance with HT enabled and disabled in test set 5.4.1.1.2 as described in Section 5.4.1.1, to analyse the impact of the HT technology on energy consumption and system performance. In these tests we kept the resources allocated to the cloud application constant and increased the $LN$ of the tasks gradually with HT on and off. In test set 5.4.1.1.3 as described in Section 5.4.1.1, we ran multiple tasks with the same $LN$ simultaneously and kept the system configuration and cloud application deployment strategy constant to analyse the scheduling overhead. Then, we kept the number of the tasks and total resource allocated to the cloud application constant while changing the cloud application deployment.
strategy in test set 5.4.1.4 as described in Section 5.4.1.1. Resource
over-commitment in cloud environments refers to allocating more virtual
resources than actual available physical resources. A virtualised system is an
overcommitted system in most cases since the total virtual CPUs are usually
more than total physical CPUs so that the resource utilisation can be
improved [106]. However, the energy consumption of the cloud system may
increase as more scheduling overhead is introduced due to resource
over-commitment. Therefore, we investigated the impact of resource
over-commitment on system performance and energy consumption in the
final test set 5.4.1.1.5 as described in Section 5.4.1.1. We kept the number of
tasks and total resources allocated to the application constant while changing
the size of VMs in this test set.

- **Memory-intensive tasks:** We deployed another cloud application containing
  one memory-intensive service unit which generates large amount of data
  without saving it using memory. The service unit consumes as much memory
  as possible based on the size of memory allocated to it. Each invocation of
  this service unit was a memory-intensive task $t_{\text{comp}}^i$. The size of processed
  file determined the workload of the memory-intensive task. In this case, the
  number of the processes for the task $PT_{\text{comp}}^i$ is number of threads in file
  processing procedure, the size of data to be processed $DT_{\text{comp}}^i$ is the size of
  the file to be processed, and the type of operation of a memory-intensive task
  $OT_{\text{comp}}^i$ is processing file in memory. We designed one sets of test cases in
  this series of experiments for memory-intensive tasks. We firstly fixed the
  resources allocated to the cloud application and increased the file size of
each task. Then we increased the resources allocated to the cloud application
while keeping the workload of each task constant. The details are presented
in test set 5.4.1.2 as described in Section 5.4.1.2.

### 5.3.2 Data-Intensive Task

There are two types of data-intensive task: file related and database related. Both of
service units required by these two types of tasks mainly consume the storage
resources in a cloud environment. The test cases designed for each type of data-intensive task are described as following:

- **File related data-intensive tasks:** This kind of data-intensive task is focused on file processing with read/write operations. We deployed a cloud application with one data-intensive service unit that calls the IOzone benchmark\(^\text{13}\) to process file in our tests. IOzone is a disk and file system benchmarking application. By calling IOzone, the service unit can generate a large number of I/O operations and stressed the disk I/O as required. Each run of the service was considered as a data-intensive task \(i\). The total amount of data read/write determined the workload of a task. In this case, the number of processes of the task \(PT_{\text{data}}^i\) is the number of threads in each file processing procedure, and the size of data to be processed \(DT_{\text{data}}^i\) is the size of the file to be processed, and the type of operation of a file related data-intensive task \(OT_{\text{data}}^i\) is processing with read/write operations on disks. In each task, we set the read/write ratio to 50% - 50%. The parameters of a file related data-intensive task to be changed include the number of threads and the size of the data record. We set the size of the data record to be transferred at 4K, 8K and 64K bytes as our basic test suite. 4K is the memory page size, 8K is what Microsoft Windows uses for network transfers, and 64K is the typical record size that Windows uses when applications try to transfer blocks of data bigger than 64K [107]. We also tested a few very large data records, including 8MB and 16MB data records. A total of five sets of tests were designed and run in this series of experiments. Test set 5.4.2.1.1 as described in Section 5.4.2.1 aimed to reveal the impact of workload on energy consumption and system throughput while keeping the task parameters and system configuration constant. The purpose of test set 5.4.2.1.2 as described in Section 5.4.2.1 was to analyse the impact of system configuration on energy consumption and system throughput. So we kept the cloud application workload and task parameters constant and changed the number and type of VMs. In test set 5.4.2.1.3 as described in Section 5.4.2.1, \(^\text{13}\) [http://www.iozone.org](http://www.iozone.org)
we aimed to evaluate the impact of task parameters on energy consumption and system throughput by keeping cloud application workload and system configuration constant. Furthermore, we evaluated the impact of scheduling overhead on energy consumption and system throughput in test set 5.4.2.1.4 as described in Section 5.4.2.1. Similar to computation-intensive tasks, resources allocated to a data-intensive service unit affects the energy consumption and system performance. Therefore, we measured the energy consumption and system throughput by changing the total resource allocated to the cloud application in test set 5.4.2.1.5 as described in Section 5.4.2.1.

- **Database related data-intensive tasks:** Another type of data-intensive task is database related which devotes most of its processing time to the movement and manipulation of data in databases. In order to investigate the system performance and energy consumption of this type of task, we deployed a cloud application with one data-intensive service unit which can query and manipulate data records on a relational database. Each run of the service was considered as a data-intensive task \( i_{\text{data}} \). In this case, the number of processes of the task \( PT_{\text{data}} \) is the number of threads in each database query requests, and the size of data to be processed \( DT_{\text{data}} \) is the size of the data block in each database query, and the type of operation of a database related data-intensive task \( OT_{\text{data}} \) is one of the database operations -“select”, “insert”, “update”, or “delete”.

In test set 5.4.2.2.1 as described in Section 5.4.2.2, we profiled the system performance and energy consumption of different operation types (select, insert, update, delete and their combinations) and data size to investigate the impact of different types of database operations on the system performance and energy consumption. In test set 5.4.2.2.2 as described in Section 5.4.2.2, we mixed all four types of database operations and kept the ratio of each type of operation constant while changing the data size of each operation, the total number of the requests and deployment strategies of the cloud application.
5.3.3 Communication-Intensive Task

A communication-intensive task in a cloud application usually generates a huge amount of network transactions between cloud user devices and cloud systems. The number of user requests and the data size of each request may impact the system performance and energy consumption. In addition, the resource allocation strategies may also impact the energy consumption of communication-intensive tasks [108]. Therefore, we profiled and analysed the system performance and energy consumption of communication-intensive tasks with different task workloads and resource allocation strategies. We deployed a cloud application with one communication-intensive service unit that took client requests of varying frequency with varying payload size and generated responses of varying size. Each run of the service was considered as a data-intensive task $t_{\text{comm}}^i$. In this case, the number of processes of the task $PT_{\text{comm}}^i$ is the number of requests for each user, and the size of data to be processes $DT_{\text{comm}}^i$ is the size of the data transmitted in each user request, and the type of operation $OT_{\text{comm}}^i$ is paging the cloud server and requesting responses. In test set 5.4.3.1 as described in Section 5.4.3, we firstly fixed the resource allocated to the cloud application while increasing the number of user requests and the packet size of each request. We then fixed the packet size of each request while changing the number of user requests per second and resource allocated to the cloud application in test set 5.4.3.2 as described in Section 5.4.3.

5.3.4 Mixed Computation- and Data-Intensive Task

Increasing computation and data processing power allows more and more scientific and business applications to be deployed in cloud environments. The workload of a scientific application is usually a mix of both computation-intensive and data-intensive tasks [109] such as Epigenome\(^{14}\). Epigenome is a scientific workflow application which maps short DNA segments collected using high-throughput gene sequencing machines to a previously constructed reference genome. The workflow reads DNA files, splits input segment files into small chunks, processes the chunks

\(^{14}\) http://epigenome.usc.edu/
and merges the mapped genome sequences into a single output map. File reading in Epigenome is a data-intensive task and small DNA chunk processing is a computation-intensive task. As a rapidly increasing number of scientific applications are moving to the cloud, it is important to investigate the system performance and energy consumption of cloud systems with such types of mixed task types. We generated the client workload of Epigenome with mixed computation-intensive and file related data-intensive tasks. We then analysed the system performance and energy consumption of the target cloud system with different workload models and cloud application deployment strategies. In test set 5.4.4.1 as described in Section 5.4.4, we kept the cloud application deployment strategy and total amount of DNA data processed constant, while changing the size of each DNA chunk. In test set 5.4.4.2 as described in Section 5.4.4, we kept the total amount of DNA data processed and size of each DNA chunk, while changing the cloud application deployment strategy.

5.3.5 Mixed Computation-, Data- and Communication-Intensive Task

Most cloud applications require a Web server to handle user requests and a database server to process the database queries in response to the user requests. Similarly, service-oriented architectures have multiple distributed compute and data services with significant inter-service communication. Therefore, a cloud application typically has workload tasks composed of a mixture of communication-intensive tasks, data-intensive tasks and computation-intensive tasks. Different user requests have different mixes of these workload task types. We aimed to investigate the impact of the application workloads and the cloud application deployment strategy on the system performance and energy consumption of the cloud system. We selected JPetStore as the cloud application to test in our experiment as it has been widely used as a representative Web application that produces a transactional workload. We generated the workload of JPetStore based on the client behaviour model introduced by Cai [110] with mixed CPU-intensive, database related data-intensive and communication-intensive tasks. We profiled and analysed the system performance and energy consumption with different application workloads and different cloud application deployment strategies. In test set 5.4.5.1 as described
in Section 5.4.5, we kept the cloud application deployment strategy constant and changed the application workload. On the contrast, in test set 5.4.5.2 as described in Section 5.4.5, we kept the workload constant and changed the cloud application deployment strategy.

5.4 Experiment Results Analysis

We conducted five major sets of tests to analyse the energy consumption and system performance of cloud-hosted application tasks in order to analyse the relationship between the energy consumption and system performance. The first three sets of tests focused on individual type of tasks and the rest two sets of the tests focused on mixed type of tasks. For each test set, we took workload and system configuration as inputs. Energy consumption of each task and the system throughput were the outputs of our experiments. We modified the configuration of SwinCloud, including the number of VMs, the hardware and software resources allocated to the VMs. We then measured the energy consumption of SwinCloud running different types of tasks. Each set of tests was repeated ten times to reduce measurement error. We analysed the impact of system configuration, task parameters and workload on the energy consumption and performance of the cloud application based on the experimental results.

5.4.1 Computation-Intensive Workloads

5.4.1.1 CPU-Intensive Workloads

A CPU-intensive task usually requires a number of threads to perform the computation in cloud environments. The total energy consumption might increase with the number of threads increasing since the increased overhead of scheduling will cause extra energy consumption. Moreover, the energy consumption of the cloud application is subject to change under different system configurations.

As described in Section 5.3.1, a cloud application with one service unit which calculates a Fibonacci sequence was deployed as a representative of a CPU-intensive task. Concurrency was applied to make sure the workload would be
distributed to all available virtual CPUs. The largest number $LN$ (see Section 5.3.1) of the Fibonacci sequence is the major factor that determined the duration of each calculation task. In order to control the execution time of each task within ten minutes, we set $LN$ from 52 to 56 in the following tests. Each CPU-intensive task took over 1 minute to complete. In order to better illustrate system throughput, we converted the value of throughput from “the total number of user interactions requested and completed successfully per second” to “the total number of user interactions requested and completed successfully per hour”.

**Test Set 5.4.1.1.1:** Keeping the number of tasks constant, while gradually increasing the number of active CPU cores allocated to the cloud application, and the largest number $LN$ of each task.

The total number of tasks was set to one. In this set of tests, the cloud application was deployed on an XLarge VM (see Table 5.1 in Section 5.2 for specification details). The server power usage is presented in Figure 5.2. We observed increasing power consumption caused by increased CPU usage. The power consumption was linear to CPU usage. The energy consumption per task is displayed in Figure 5.3. We noticed that the energy consumption of the task was impacted by the number of CPU cores allocated to the cloud application. Moreover, the largest $LN$ of the Fibonacci sequence also affected the energy consumption. As shown in Figure 5.3, the energy consumption of each task increased with the workload of the task. Moreover, the energy consumption of each task decreased as the number of cores allocated to the cloud application increased. This is because the execution time of a task will decrease as more computation resources are allocated to the cloud application. However, the increase in average energy usage rate caused by extra CPU cores is not as much as the execution time of the task. Therefore, the energy consumption will decrease accordingly. In addition, we observed a slight turning point of the energy consumption when the number of cores allocated to the application reached three. For instance, when we set the largest number of the Fibonacci sequence $LN$ to 56, the energy consumption with four cores allocated increased by 4% compared to energy consumption with three cores. This shows that the overhead of scheduling an extra core can cancel out the task’s running time saved and will also cause more energy consumption. Therefore, the energy consumption as a function of number of
cores is highly nonlinear, with a minimum at three. In addition, we observed that most reduction of energy consumption is in changing from one core to two cores, and then it does not change much anymore. The system throughput is presented in Figure 5.4. As expected, the more resources allocated to the cloud application the better the resulting system throughput. This result shows that, for CPU-intensive tasks, the system throughput rises with the number of cores allocated to the cloud application and the increase of system throughput is nonlinear. Together with the results of Figure 5.3, we can conclude that energy consumption of CPU-intensive tasks is a more complex nonlinear function of cores allocated to the cloud application and execution time.

Figure 5.2 Server Power Consumption with Different Workload

Figure 5.3 Energy Consumption per Task with Different Workload
Test Set 5.4.1.1.2: Keeping the number of tasks and resources allocated to the cloud application constant, gradually increasing the largest number $LN$ of each task, and turning Hyper Threading on and off.

Only one task was running on the test-bed at one time. In this set of tests, the cloud application was deployed on an XLarge VM. The results are presented in Figure 5.5 and Figure 5.6. As shown in Figure 5.5, with Hyper-Threading enabled, there is a significant increase in the energy consumption of the task – the Hyper-Threading causing extra power consumption. In the case of “$LN = 56$”, the extra energy consumption caused by Hyper-Threading is 8%. However, as shown in Figure 5.6, the extra system throughput obtained by enabling Hyper-Threading is not as much as expected. In the cases of “$LN = 53$”, “$LN = 54$” and “$LN = 55$”, Hyper-Threading even led to decrease in system throughput. This suggests that Hyper-Threading be disabled for CPU-intensive tasks since there is little to gain form Hyper-Threading if a core’s execution resources are already well utilised.
Test Set 5.4.1.1.3: Keeping the largest number $LN$ of each task constant, and increasing the total number of tasks and number of VMs allocated to the cloud application.

In this set of tests, the type of each VM was set to Small. Figure 5.7 shows the results. In order to illustrate small differences in energy consumption data with different number of VMs allocation to the cloud application, the value of energy consumption per task starts from 3300 instead of 0 in Figure 5.7. The number of running tasks can influence the average energy consumption of a single task: a larger number of tasks lead to higher average energy consumption of individual tasks. When the total number of tasks increased, more consequential scheduling overhead caused extra energy consumption. Moreover, it is clear that the increase in number of VMs significantly resulted in increase in per-task energy consumption. For instance, in the case of 6 tasks, the per-task energy consumption increased by 5% when we increased the number of VMs from one to two. As more VMs were deployed, the VM maintenance overhead increased as well, resulting in more power consumption. On the other hand, as depicted in Figure 5.8, the throughput was minimal when only one VM was configured. The throughput of two VMs and three VMs were at the same level. However, three VMs consumed 3% more energy than two VMs on average, as presented in Figure 5.7. Thus the two VMs configuration demonstrates higher energy efficiency than other configurations when there are multiple CPU-intensive tasks running simultaneously.
Test Set 5.4.1.1.4: Keeping the number of tasks and resource allocated to the cloud application constant, while changing the cloud application deployment strategy.

The total number of tasks was set to 16. In this set of tests, the cloud application was deployed on one XLarge VM, two Large VMs and four Small VMs respectively. In order to control the execution time of each task, we set the largest number of the Fibonacci sequence LN from 46 to 49. All tasks were evenly distributed on all the VMs. The results of performance and energy consumption are presented in Figure 5.9 and Figure 5.10. The energy consumption per task and throughput were at the same level under different cloud application deployment strategies. However, the energy consumption and throughput increased slightly when we changed the cloud application deployment strategy from one XLarge VM to four Small VMs. For instance, when we set the largest number of the Fibonacci sequence LN to 46, the energy consumption increased by 1.1% and throughput increased by 1.1% with four Small VMs compared to energy consumption and throughput with an XLarge VM. The more VMs configured, the more scheduling
overhead was introduced. Therefore, energy consumption was higher. On the other hand, more VMs configured make all the running tasks take full advantage of other resources such as memory. Thus, throughput improved.

![Figure 5.9 Energy Consumption per Task with Different Deployment Strategies](image1)

![Figure 5.10 Throughput with Different Deployment Strategies](image2)

Test Set 5.4.1.1.5: Keeping the number of tasks constant, while changing the resources allocated to a cloud application with none-overcommitted and overcommitted strategy.

The total number of tasks was set to 16. In this set of tests, the cloud application was deployed on two Medium VM, two Large VMs and two XLarge VMs respectively. In order to control the execution time of each task, we set the largest number of the Fibonacci sequence LN from 46 to 49. All tasks were evenly distributed on all the VMs. The results of performance and energy consumption are presented in Figure 5.11 and Figure 5.12, respectively. As introduced in Section 5.2.1, four physical CPUs are available on each SwinCloud server. When we deployed the cloud application on two Medium VMs, the number of virtual CPUs was equal to the number of physical CPUs. When we deployed the cloud application on two
Large VMs and two XLarge VMs, the total number of virtual CPUs was six and eight, respectively. Thus, the computation resources were overcommitted as the total number of virtual CPUs allocated exceeded the total number of physical CPUs. Energy consumption per task increased and system throughput decreased when we changed the resource allocation strategy from the none-overcommitted case to the overcommitted case as presented in Figure 5.11 and Figure 5.12, respectively. CPU virtualisation adds a certain amount of overhead. In addition, overcommitted resource allocation introduces synchronisation overheads of multiprocessors on the physical server. Both of the abovementioned overheads will result in performance degradation. Therefore, the execution time of each task increased, system throughput decreased and the energy consumption of each task increased accordingly. Overcommitted resources allocation should be avoided in order to improve the energy efficiency of cloud systems.

![Figure 5.11 Energy Consumption per Task with Different Resource Allocation Strategies](image1)

![Figure 5.12 Throughput with Different Resource Allocation Strategies](image2)
5.4.1.2 Memory-Intensive Workloads

As introduced in Section 5.3.1, a file processing cloud application with one service unit was deployed to process memory-intensive tasks. The application consumes as much memory as possible based on the size of memory allocated to it. The size of processed file is the major factor that determined the execution time of a memory-intensive task. Similar to CPU-intensive tasks, we converted the value of throughput from “the total number of user interactions requested and completed successfully per second” to “the total number of user interactions requested and completed successfully per hour” in order to better present system throughput in figures.

Test Set 5.4.1.2: Keeping the number of tasks constant, while gradually increasing the size of RAM allocated to the cloud application, and the file size processed by each task.

The total number of tasks was set to 10. In this set of tests, the cloud application was initially deployed on a Small VM. We then gradually increased the size of RAM configured on the VM in the test. With each RAM configuration, the size of file processed was set to 10GB, 15GB, 20GB and 25GB, respectively. The server power consumption and average memory usage are presented in Figure 5.13 and Figure 5.14. When we increased total memory allocated to the cloud application from 2GB to 8GB, the average memory usage of the server increased accordingly as showed in Figure 5.14. However, the power consumption of the server remained at the same level as displayed in Figure 5.13. Task memory usage has only slight impacted on total power consumption. Other research on the power consumption of memory reports that the power consumption of memory remains constant regardless of the workloads. However, power consumption of memory is proportional to the number of memory chips [111]. In addition, the execution time of a task will increase when we increase the size of file processed by the task. Therefore, the energy consumption of each task will increase and the system throughput will decrease, as presented in Figure 5.15 and Figure 5.16.
Figure 5.13 Server Power Consumption with Different File Size

Figure 5.14 Server Memory Usage with Different File Size

Figure 5.15 Energy Consumption per Task with Different File Size

Figure 5.16 Throughput with Different File Size
5.4.2 Data-Intensive Workloads

5.4.2.1 File Related Data-Intensive Workloads

A file related data-intensive task needs to retrieve or store a large amount of data stored in one or more data storage servers. It requires high local disk I/O bandwidth in order to meet performance requirements. In reality the storage servers can be deployed in different data centres located in different geographic locations. However, we only consider the energy consumption in one data centre for the purpose of simplicity as the characteristics of data centres of the same service provider are often very similar.

The amount of data transferred in and out of the storage server determined the execution time of a file related data-intensive task. We focused on the correlation between the energy consumption and the amount of data transferred in and out of the storage server. To profile and analyse the energy consumption caused by different data sizes and system configurations, IOzone was deployed as the cloud application to process file related data-intensive tasks which is introduced in Section 5.3.2.

In every set of tests, we observed major spikes of energy consumption at the beginning and at the end of each task. They corresponded to the moments when disk I/O were stressed. We conducted five sets of tests in this series of experiments.

Test Set 5.4.2.1.1: Keeping the total number of tasks, process number of each task, resources allocated to the cloud application and VM configuration constant, while increasing the total amount of data transferred and record size.

In this set of tests, the cloud application was deployed on an XLarge VM. Only one task was running in at one time and the number of processes was fixed at one. Therefore, the impact of scheduling overhead caused by multiple tasks and processes on energy consumption were avoided. The power consumption of the server is illustrated in Figure 5.17. In order to present the small differences in the server power consumption with different record size, the value of server power consumption starts from 90 instead of 0. As presented in Figure 5.17, server power consumption increased as the data record size decreased. However, the effect of record sizes on power consumption is nonlinear as displayed in Figure 5.17, with larger record sizes leading to convergence to a certain value. For instance, server
power consumption did not change too much when we increased the record size from 8MB to 16MB. Smaller data record sizes led to generation of more overhead information for keeping track of where the data were located on the storage media. The overhead information consisted of the directory information, the space allocation and any other data that was not a part of the data to be transferred. As a result, the tasks used more computational resources, consuming more energy.

![Figure 5.17 Server Power Consumption with Different File Size](image)

**Figure 5.17 Server Power Consumption with Different File Size**

The results of system energy consumption and system throughput are displayed in Figure 5.18 and Figure 5.19, respectively. From Figure 5.18, we see that the energy consumption increased proportionally to the total amount of transferred data. The per-task energy consumption increased in a linear manner. As the total amount of data transferred increased, smaller record size would cause higher gradient when the total energy consumption increased because the execution time and the energy consumption increased simultaneously. As the record size decreased, more overhead information needed to be processed and stored. In summary, if only one process is configured to run the data-intensive task, bigger record size is more energy-efficient.

As shown in Figure 5.19, the system throughput was higher when we set the record size to 64K, compared to other record sizes. As 64K is the typical record size Windows uses when applications try to transfer blocks of data bigger than 64K, data is transferred most efficiently in this size.
Test Set 5.4.2.1.2: Keeping total number of tasks, process number per task, total amount of data transferred and resource allocated to the cloud application constant, while changing VM configurations and record size of each task.

In this set of experiments, the cloud application was deployed on one XLarge VM, two Medium VMs and four Small VMs respectively so as to fix the size of the total virtual memory and the total number of virtual cores allocated to the VMs. The total number of tasks was set to four and the number of processes instantiated for each task was set to one. All the tasks were distributed on the VMs evenly. The total amount of data to be transferred was 64GB.

The system energy consumption and system throughput are presented in Figure 5.20 and Figure 5.21. We observed that the energy consumption increased and the throughput decreased as we changed the cloud application deployment from one XLarge VM to four Small VMs. On one hand, VMs compete with each other for I/O bandwidth. On the other hand, more VMs cause more read/write latency that lead to longer execution time of each task. As shown in Figure 5.20, the per-task energy
consumption increased in a sub-linear manner. As shown in Figure 5.21, when we ran the workload on one XLarge VM, the system throughput was the highest. From these tests we conclude that running one VM for the entire I/O operation is most energy-efficient.

![Figure 5.20 Energy Consumption with Different VM Configurations](image)

**Figure 5.20 Energy Consumption with Different VM Configurations**

**Figure 5.21 Throughput with Different VM Configurations**

**Test Set 5.4.2.1.3:** With single task, keeping the total amount of data transferred stable, resource allocated to the cloud application and VM configuration constant while increasing the process numbers for task and the record size of the task.

The cloud application was deployed on an XLarge VM to run this set of tests. Only one task was running and it was assigned to transfer a 64GB file. As depicted in Figure 5.22, the per-task energy consumption increased with the number of processes. Although the power consumption of the server decreased slightly as the number of processes increased, as presented by Figure 5.23, multiple processes caused scheduling and synchronisation overhead. Thus, the total execution time of one data-intensive task increased accordingly. As a result, the system throughput
decreased, as shown in Figure 5.24. Our conclusion is that running data-intensive tasks with one sequential I/O operation is the most energy-efficient when the total amount of data transferred is fixed.
Test Set 5.4.2.1.4: With multiple tasks, fix the total amount of data transferred, the resource allocated to the cloud application and number of processes for each task, changing VM configuration and record size for each task.

The cloud application was deployed on one XLarge VM, two Medium VMs and four Small VMs, respectively. The total number of tasks was set to four and the tasks were distributed to all the VMs evenly. We fixed the number of processes for each task at four and the total amount of data to be transferred of all the tasks at 64GB. The system energy consumption and system throughput are shown in Figure 5.25 and Figure 5.26, respectively. As we already observed in previous tests, energy consumption is greatly impacted by data record size. Smaller record size results in more energy consumption. The highest throughput was approximately 30% less than the highest throughput we obtained in test set 5.4.2.1.2. The numbers of process in both test sets were the same. However, there were multiple tasks running simultaneously this test set while only one task was running in test set 5.4.1.1.2. Therefore, multiple task scheduling causes extra scheduling overhead.

![Figure 5.25 Energy Consumption with Multiple Processes and Multiple VMs](image1)

![Figure 5.26 Throughput with Multiple Processes and Multiple VMs](image2)
Test Set 5.4.2.1.5: Keeping the total number of tasks, the total amount of data transferred and the number of processes for each task constant, while changing the resources allocated to the cloud application, VM configuration and record size of each task.

Only one task was running on the cloud test-bed at one time. We fixed the number of processes for the data-intensive task at one and the total amount of data transferred at 64GB and scaled up the size of VM allocated to the cloud application from Small to XLarge. These results are shown in Figure 5.27 and Figure 5.28. Compared to computation-intensive tasks, the VM type did not impact the energy consumption and throughput significantly when the record size was fixed. Running on an XLarge VM was most energy-efficient when the record size was smaller than 64K. When the record sizes were 8M and 16M, running the data-intensive task on a Small VM was most energy-efficient.

![Figure 5.27 Energy Consumption per Task with Different VMs](image1)

![Figure 5.28 Throughput with Different VMs](image2)
5.4.2.2 Database Related Data-Intensive Workloads

A database related data-intensive task in cloud environment usually involves processing and manipulating large amounts of data to and from database storage. We deployed a cloud application with one service unit that can query and manipulate data records in a relational database to process data-intensive tasks as introduced in Section 5.3.2. We selected mixes of database operations, “select”, “insert”, “update” and “delete”. The size of data processed by the database operations determined the execution time of the data-intensive task.

Test Set 5.4.2.2.1: Keeping the number of tasks, and resource allocated to the cloud application constant, while gradually increasing the size of data processed of each task.

The total number of tasks was set to 1000 and user request rate was set to 10 per second. The cloud application was deployed on an XLarge VM. SQL server 2005 was installed to process all the database requests. The result of energy consumption is presented in Figure 5.29. System throughput and response time are presented in Figure 5.30 and Figure 5.31 respectively. As illustrated in Figure 5.29, energy consumption of “insert” and “update” operations increased dramatically when we increased the record size of each request. However, the energy consumption of “delete” and “select” operations only had a slight increase. The “insert” and “update” operations both require reading and writing large amount of data on the disk compared to “select” and “delete” operations, which results in more power consumption of the server. In addition, the response time of “insert” and “update” operations were much longer than “delete” and “select” operations. Therefore, task execution time of “insert” and “update” operations increased, which caused high energy consumption and low throughput as displayed in Figure 5.30.
A research on relational database workload characterisation reports that the ratio of “select”, “insert”, “update” and “delete” in database workload are 75.86%, 4.69%, 7.75% and 11.69% respectively [112]. We adopted the abovementioned ratio of each database operation in our tests. The total number of tasks was set to 1000. We gradually increased the user request rate from 10 to 40. In this set of tests, the
The record size of each database request was set to 400KB and 500KB respectively. The cloud application was deployed on an XLarge VM in this set of tests. SQL server 2005 was installed with default configurations to process all the database requests. The system performance and energy consumption are presented in Figure 5.32 and Figure 5.33. As displayed in Figure 5.33, the throughput decreased slightly while increasing the record size from 400KB to 500KB. This is because response time increased as the record size increased as displayed in Figure 5.34. Therefore, the task execution time increased and throughput decreased accordingly. As presented in Figure 5.32, the energy consumption of all the tasks increased dramatically as record size increased. Total amount of data read and write on hard disk increased as the record size increased, which resulted in longer task execution time. In addition, the bigger record size introduced more data reading/writing scheduling overhead and the power consumption of the server increased. Therefore, the energy consumption increased. As presented in Figure 5.32 and Figure 5.33, the energy consumption decreased and throughput increased when we increased the number of user requests per second. However, there was a turning point when the number of user requests per second reached 30. For instance, when we set the record size to 500KB and user requests per second to 40, the energy consumption increased 7.8% and the throughput decreased 3.2%. This is because task consolidation will increase the resource utilisation which will reduce the total execution time. However, when the user requests reach 40 per second, the overhead of scheduling and synchronising user requests can cancel out the task running time saved and will cause more energy consumption.

**Figure 5.32 Energy Consumption per Task with Different Record Sizes**
5.4.3 Communication-Intensive Workloads

A communication-intensive task in cloud environments usually generates a huge amount of network transactions between cloud user devices and cloud systems. It requires a lot of network resources to transmit large amounts of data. Switches form the basis of the interconnection fabric of a private cloud network. Thus, switches, plus network cards, are the main energy consumers of the network resources. Traditionally, energy consumption of a switch depends on hardware parameters, such as the type of switch, the number of ports and the port transmission rates. However, the energy consumption may increase with the amount of data transferred over the network because of the data processing overhead. In addition, the total energy consumption might be impacted by network congestion because of the imbalance between the computation speed and the communication speed. The computation speed of the switch may be slower than the data transmission speed. We investigated this issue by applying different network workloads. A Cisco 2960 switch connected the cloud servers and client PCs. All parameters of the switch were
set as default to avoid potential impact caused by switch configuration. We deployed a cloud application with one service unit that handled user requests and generated responses upon the receipt of user requests as communication-intensive task which is introduced in Section 5.3.3. The total amount of user requests and the size of data transmitted in each user requests determined the execution time of a communication-intensive task.

In order to reveal the impact of the network packet size, the types of user requests and the system configuration on the energy consumption of communication-intensive tasks, we conducted two sets of tests in this series of experiments, described as follows:

**Test 5.4.3.1:** *Keep the resource allocation strategy of the cloud application constant, while increasing the number of user requests and the packet size of each request.*

The cloud application was deployed on a Small VM in this set of tests. We increased the user requests per second from 300 to 1200 in steps of 300. The packet size of each user request was increased from 1500KB to 2500KB in steps of 500. The results of energy consumption and throughput are presented in Figure 5.35 and Figure 5.36. As presented in Figure 5.35, there was an increase in the energy consumption of the task when we increased the packet size. For instance, when we set the user requests per second to 300, the energy increased 36.7% when we increase the packet size from 1500KB to 2500KB. Furthermore, the throughput decreased as the packet size increased. Bigger packet size usually leads to more transmission time over the network and more processing time in the cloud environment. Accordingly, throughput decreases and energy consumption increases for the communication-intensive task.
Test Set 5.4.3.2: Keep the packet size of each request constant, while changing the number of user request per second and resource allocation strategy of the cloud application.

The packet size was set to 2500KB. The results of energy consumption and throughput are presented in Figure 5.37 and Figure 5.38. When we increased the VM allocated to the cloud application from Small to XLarge, the energy consumption decreased while system throughput increased in general. Intuitively, the more resources used the greater the energy consumption. However in this case, the smaller the instance the higher the disk accesses due to the thrashing of the cache, which leads to increase in energy consumption. Noticeably, when the size of the VM changed from Large to XLarge, the system throughput decreased and the system energy consumption increased in general. When we set the type of VM to XLarge, the total capacity of the VM reached the full capacity of the physical server. The resources left for VM management were less, which led to longer processing time of each user request. Therefore, deploying the cloud application on a Large VM is the most energy efficient.
Mixed Computation- and Data-Intensive Workloads

Increasing computation and data processing power allows more and more scientific and business applications to be deployed in cloud environments. A scientific application is both computation-intensive and data-intensive, where computed and retrieved data sets from the cloud data centre are often gigabytes or even terabytes.

We selected a DNA processing application named Epigenome as our representative scientific application which included two service units – one CPU-intensive and one data-intensive. We generated the workload of Epigenome with 50% CPU-intensive and 50% file related data-intensive tasks. Firstly, a CPU-intensive task and a file related data-intensive task are executed sequentially. Then the process is repeated until all data have been processed. Two sets of tests were executed in total. The details of the experiment results are presented as follows:

Test Set 5.4.4.1: Keep the resource allocation and deployment strategy of the cloud application, and total amount of data processed constant, while changing the size of each data set.
The cloud application of the scientific application was deployed on an XLarge VM in this set of test. We set the total amount of data processed to 2GB. We increased the data set size of the data-intensive task from 1000KB to 8000KB. The computation-intensive task scale $LN$ was increased from 36 to 39. The results of energy consumption and system throughput are presented in Figure 5.39 and Figure 5.40. From Figure 5.39, we can see that the application energy consumption decreased when we increased the size of the data set in a linear manner. As the size of the data set increased, less overhead information needed to be processed and stored, which led to shorter execution time of the same workload. Thus, system throughput increased, shown in Figure 5.40.

![Figure 5.39 Energy Consumption with Different Sizes of Data Block](image1)

Test Set 5.4.4.2: Keep the total amount of data processed and size of each data set constant, while changing the resource allocation and deployment strategy of the cloud application.

In this set of test, the data set size of the data-intensive task was set to 8000KB and the task scale $LN$ of computation-intensive task was set to 39. Firstly, we
deployed both service units of the scientific application on one XLarge. Then, we deployed the scientific application on two Medium VMs and four Small VMs respectively with computation-intensive service unit and data-intensive service unit deployed on the same VM. We named the deployment strategies “2Medium(S)” and “4Small(S)” respectively. The workload was evenly distributed on all the VMs. Finally, we deployed the scientific application to two Medium VMs and four Small VMs respectively with computation-intensive service unit and data-intensive service unit on different VMs. We named the deployment strategies “2Medium(D)” and “4Small(D)” respectively. The workload was also evenly distributed on all the VMs. The results of application energy consumption and system throughput are presented in Figure 5.41 and Figure 5.42. As displayed in Figure 5.41, the energy consumption of the application varied with different service units deployment strategies. When we deployed the application on the VMs with the same scale, the energy consumption increased when we changed the deployment strategy of the computation-intensive service unit and data-intensive service unit from the same VM to different VMs. In contrary, the system throughput increased as displayed in Figure 5.42. For instance, when we change the deployment strategy from “2Medium(S)” to “2Medium(D)”, the energy consumption increased 33.5% while throughput decreased 40.3%. This is because deploying the computation-intensive service and data-intensive service on different VMs will introduce more communication overhead between VMs, which will result in more processing time. In addition, when we increase the number of VMs, the energy consumption increased and system throughput decreased no matter how the two kinds of service units were deployed (on the same VM or different VMs). For instance, when we change the deployment strategy from “2Medium(S)” to “4Small(S)”, the energy consumption increased 4.5% and throughput decreased 5.8%. This is because more VMs will introduce extra operation system scheduling overhead, which will cause longer service requests processing time of the cloud application workload. The more VMs are allocated to the cloud application, the more concurrent processes are created to process the service requests of the cloud application. However, the extra service requests processing time introduced by the extra VM operation system scheduling overhead cannot be cancelled out by the newly created concurrent processes. The overall application execution time will be longer. Therefore, energy consumption will increase and throughput will decrease.
accordingly. In summary, deploying the scientific application on two Medium VMs with all kinds of service units on the same VM is the most energy efficient while achieving the best system performance.

![Figure 5.41 Energy Consumption with Different Deployment Strategies](image1)

**Figure 5.41 Energy Consumption with Different Deployment Strategies**

![Figure 5.42 Throughput with Different Deployment Strategies](image2)

**Figure 5.42 Throughput with Different Deployment Strategies**

### 5.4.5 Mixed Computation-, Data- and Communication-Intensive Workloads

Most cloud applications have workload tasks composed of a mix of communication-intensive tasks, data-intensive tasks and computation-intensive tasks. Different user requests have different mixes of these workload task types. JPetStore was selected as the cloud application to test in our experiment as it has been widely used as a representative Web application that produces a transactional workload. It contains three major types of service units: CPU-intensive, database related data-intensive and communication-intensive. We generated the workload of JPetStore based on the client behaviour model introduced by Cai [110] mixed with CPU-intensive, database related data-intensive and communication-intensive tasks.
Two sets of tests were conducted in total. The details of the experiment results are presented as follows:

**Test Set 5.4.5.1: Keep the resource allocation strategy of the cloud application constant while changing workload.**

This cloud application was deployed on a Large VM in this set of test. The initial number of users was set to 10. We increased the concurrent requests number of each user from 50 to 200 in steps of 50. The results of energy consumption and system throughput are presented in Figure 5.43 and Figure 5.44. As the number of concurrent requests increased, the energy consumption increased as displayed in Figure 5.43. The throughput decreased accordingly as shown in Figure 5.44. In order to illustrate the small difference in system throughput with different number of user requests, the value of system throughput starts from 90 instead of 0 in Figure 5.44. Intuitively, more user requests will introduce more scheduling and synchronising overhead in the cloud application, which will result in increase of the processing time of each user request.

![Figure 5.43 Energy Consumption with Different Numbers of User Requests](image1)

![Figure 5.44 Throughput with Different Numbers of User Requests](image2)
Test Set 5.4.5.2: Keep the workload constant while changing the resource allocation strategy of the cloud application.

The initial number of users was set to 10 and the concurrent user requests of each user were set to 100 in this set of tests. We firstly deployed the cloud application on one Large VM. Then we deployed the cloud application on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on different VMs respectively, named “3Small(D)”. Finally, we deployed the cloud application on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on same VMs respectively, named “3Small(S)”. The application workload was evenly distributed on all three VMs with “3Small(S)”. The energy consumption and system throughput are presented in Figure 5.45 and Figure 5.46. Although the total resources such as CPU and RAM allocated were the same, the energy consumption decreased when deploying the cloud application on multiple VMs compared to deploying the cloud application on single VM as shown in Figure 5.45. The system throughput increased accordingly as displayed in Figure 5.46. In order to illustrate the small difference in energy consumption and system throughput, the vertical axis in Figure 5.45 and Figure 5.46 start from 5600 and 111 respectively. When deploying the cloud application on multiple VMs, the service requests of the cloud application were processed in more concurrent processes, which reduced the execution time of the cloud application. In addition, we observed that the energy consumption with deployment strategy “3Small(S)” increased 0.8% compared to “3Small(D)”. The system throughput of “3Small(S)” decreased 2.1% compared to “3Small(D)”. This is because in the workload we have modelled in this test, the majority of all the tasks are communication-intensive. Deploying communication-intensive service unit on one single VM greatly reduces the overhead of concurrent processes between different VMs. In summary, deploying this cloud application on three Small VMs and separating different types of cloud services units on different VM is most energy efficient while achieving the best system performance.
5.5 Discussion and Conclusion

Based on our observations, we have validated that system performance and energy consumption of cloud applications are highly coupled with the task parameters and system configurations as specified in our energy consumption model. Therefore, these factors should be taken into consideration when analysing the trade-off between system performance and energy consumptions of different deployment strategies for cloud applications. Our experiment results show the relationship between energy consumption, system configuration and workload, as well as system performance of cloud applications. We also derived a set of guidelines based on our experimental results. These guidelines can be adopted to achieve energy efficient deployment, resource provisioning and management strategies for cloud applications. These guidelines are presented as follows:

1. The more resources used by a single task, the more energy consumes, and the better the system performance tends to be. However, based on the type of the
runtime task, the declining throughput results show that overcommitted resource allocation results in significant increase in energy consumption and decrease in overall system performance. For instance, in test set 5.4.1.1.5 presented in Section 5.4.1.1, when we set the largest $LN$ of the Fibonacci sequence to 49 and changed the resource allocation strategy from two Medium VMs to two Large VMs, energy consumption per task increased by 7.2% on average and the overall system performance decreased by 1.3%. Therefore, we should avoid overcommitted resource allocation when deploying a cloud application in cloud environments. In addition, the resource allocation should dynamically adapt to the customers’ needs, taking both performance (and other) SLAs and energy efficiency needs into account. Dynamic scaling-up is needed when the peak workload is likely to exceed the capacity of the cloud system. Based on our results, it is worth finding the trade-off between the energy consumption caused by the overcommitted resource allocation and the energy consumption introduced by adding new resources in cloud environments.

2. The types and workload of runtime tasks impact energy consumption significantly. The energy consumption of each task is highly coupled with the resource utilisation in cloud environments. Thus, it is important to predict the required resources accurately based on the types and workloads of runtime tasks. For some applications, their workload is either known or can be empirically determined and is relatively constant. However, due to the dynamic nature of many cloud applications and the demand of different hosting platforms, the workload of different runtime tasks in cloud environments can drastically change over time. The need to find out the workload patterns for different runtime tasks in cloud environments, in order to schedule them for optimal performance and energy consumption, has emerged.

3. For a specific type of task, the various configuration parameters associated with the task, that is, the number of processes $PT$, the size of data to be processed $DT$, and the type of operation it requires $OT$, can greatly affect the task’s energy consumption. These task parameters, that may originally come from application requirements, are closely linked to system configurations $C$. 

96
Based on our observation, even with the same resource allocation of the cloud application, different system configurations such as attributes of the resources can result in different energy consumption. Therefore, dynamically changing cloud systems configurations is needed to adapt to different tasks based on their various configuration parameters.

4. For a specific cloud application, the deployment strategy can greatly affect the energy consumption and system performance. For instance, deploying the scientific application on two Medium VMs with a computation-intensive service unit and a data-intensive service unit isolated on two different VMs is the most energy efficient. On the contrary, when we deploy the JPetStore to three Small VMs with all tasks evenly distributed on all the VMs is the most energy efficient. As discussed, both of the abovementioned deployment strategies result in the reducing communication and scheduling overhead inside the deployed cloud application. Therefore, it is important to avoid communication overhead within the cloud application when deciding deployment options.

5. Differing task types, task parameters, and cloud platform configurations can dramatically affect task throughput performance and energy consumption. As discussed, some of these effects are predictable, while others are counter-intuitive. Certain configurations give optimal balance of maximising workload vs minimising energy consumption. However, there may still be times that customers and/or cloud providers choose to prioritise one over the other, that is, to sacrifice performance to maximise energy efficiency further, or to sacrifice energy efficiency for improved performance.

The major limitation of the experiments that we conducted in this chapter is that we only verified the parameters in our energy consumption model which affect the energy consumption of cloud applications. However, different tasks scheduling algorithms applied in the target cloud environment also has significant impact on energy consumption of cloud applications [113]. Task scheduling dispatches the task submitted by cloud users to available resources in cloud environments. Different scheduling algorithms result in different CPU utilization, task completion time, etc. Thus, the energy consumption of cloud applications is various based on different
scheduling algorithms. In our experiments, we adopted the default task scheduling method which was First Come First Serve (FCFS) in the cloud environment. We left research to determine the energy consumption incurred by different task scheduling algorithms for future work.

5.6 Summary

In this chapter, we presented the experiments that we have conducted to profile and analysed the system performance and energy consumption in cloud environments based on both individual type of tasks and mixed types of tasks. We collected fine-grained system performance and energy consumption data with varying workloads and system configurations. Our experimental results showed that the system performance and energy consumption of cloud applications are related to cloud system configurations and application workloads. Specifically, we first introduced the experiment setup, including test-bed and energy consumption profiling method. Then we discussed the details of test case design. Furthermore, we analysed and discussed experiment results for each set of experiment in detail. At last, we presented the guidelines of system performance and energy consumption we derived from the experimental results analysis.
Chapter 6

Energy Efficiency Analysis of Cloud Applications using Formalised Signatures

Due to the variety of cloud application workloads and cloud system configurations, it is time consuming to perform load test and to collect performance and energy data in all scenarios. In order to reduce the time and effort of conducting system performance and energy consumption trade-off analysis, we extracted a number of system performance and energy consumption patterns from the experimental results presented in Chapter 5. These patterns attempt to codify common energy efficient application deployment and configuration scenarios. Performance engineers or system architects can use these patterns as guidelines in assessing the energy efficiency of cloud applications with a given workload and application deployment strategy. The patterns are formalised as “signatures”. Each signature specifies a set of invariants that indicate the likely system performance and energy consumption of the cloud system under a given energy efficiency analysis scenario.

This chapter is organised as follows. Section 6.1 gives an overview of our approach. Section 6.2 introduces the energy efficiency pattern definition schema. Section 6.3 presents the formalised energy efficiency analysis signatures. We discuss key advantages and limitations of our approach in Section 6.4. Finally, Section 6.5 summarises this chapter.

6.1 Introduction

Many research efforts have been made to profile and analyse the energy consumption of cloud systems by running different cloud applications on different cloud platforms [4, 14, 15, 18]. A series of basic energy consumption patterns of
cloud systems has been identified based on the profiling and analysing results. They focused on analysing system performance and energy consumption of specific hardware type. In particular, they conclude the power consumption of CPU is almost linear to the CPU usage and CPU frequency scaling [13, 114-116]. For instance, Chen et al. [18] developed a linear power model that presents the behavior and power consumption of individual hardware components of a single physical server.

A key weakness of these approaches is their lack of formalisation. Evaluating system performance and energy consumption of cloud applications at early stage without conducting any load test helps reduce effort and accelerate the evaluation process. However, due to the heterogeneity of cloud platforms and cloud application workloads, these informal patterns cannot be blindly applied to predict the likely system performance and energy consumption of cloud applications running on a different cloud platform. Ideally, the system performance and energy consumption patterns should be extensible and reusable to support the capture of new system performance and energy consumption patterns on different cloud platforms.

In addition, most of the abovementioned profiling and analysis focus on the energy consumption of individual system components at runtime, such as CPU, cache, hard disk and memory. Some research aims to link the utilisation of each hardware component to its energy consumption. For instance, Joulemeter, a power meter for VMs [23], makes use of software components to monitor the resource usage of VMs and then converts the resource usage to energy consumption based on the power model of each individual hardware component. Unfortunately, performance engineers or system architects still need to run actual empirical load test experiments in order to obtain the resource usage of the VMs. This kind of approach also cannot be applied in predicting the trend of system performance and energy consumption. In addition, the abovementioned patterns and power models simply take one aspect of the load test scenarios when analysing system performance and energy consumption, i.e. cloud application workload or system configuration, into account. However, cloud application deployment strategies also have an impact on system performance and energy consumption of cloud applications, as discussed in Chapter 5.
For example, as presented in Section 5.4.5, a cloud application named JPetStore was deployed on one Large VM. Then it was deployed on three small VMs with workload evenly distributed on all three VMs. By comparing the system performance and energy consumption under two different deployment strategies, we found that deploying JPetStore on three Small VMs is more energy efficient compared to deploying it on one Large VM. Therefore, system performance and energy consumption patterns that can also capture the details of application deployment strategies in load test scenarios are necessary. These patterns will be used to predict the likely system performance and energy consumption of a cloud application without having to run extensive empirical load test.

To address the abovementioned issues, we have developed a new system performance and energy consumption pattern definition schema. This schema captures the details of a given system performance and energy consumption evaluation scenario, including cloud application workload, cloud system configurations, required resources, application deployment strategies, etc.

A key entry of the energy efficiency analysis schema is the system performance and energy consumption signature. This signature specifies a set of invariants that, when matched, indicates that the likely system performance and energy consumption to the given energy efficiency analysis scenario. We adopt the declarative and formal Object Constraint Language (OCL) [88], a well-known and extensible formal specification language to capture such signatures. This makes it easier for performance engineers or system architects to adopt the signatures in their own load test scenarios for evaluating the energy efficiency of cloud applications.

We have developed a workload model to capture the details of cloud applications’ workloads. An architecture model has been developed to describe the architectures and system configurations of cloud systems. These two models are inspired from our test case design and result analysis of the experiments described in Chapter 5. The cloud system architecture model captures the main elements in a cloud system, including data centres, physical servers, VMs, CPUs, RAM, disk, etc. Each element has a set of attributes. For instance, a physical server has attributes such as name, location, number of VMs, etc. The workload model captures the user behaviours of cloud users, including the actions of the users and the data processed.
by each action. The application workload model and cloud system architecture model provide the foundation for formalising of the energy consumption patterns as “signatures” using OCL.

6.2 Energy Efficiency Pattern Definition Schema

Extensive experiments have been conducted to profile and analyse the energy efficiency of cloud applications by applying heterogeneous workload on different VMs, as presented in Chapter 5. We have identified the key factors - e.g. cloud application workload, cloud system configurations, cloud application deployment strategy - that can impact the energy efficiency of cloud applications in these experiments. Thus, these key factors need to be specified in our energy efficiency patterns. The details of these key factors are described as following:

**Application Type**: A cloud application consists of several functional service units. Each service unit utilises different resources in the cloud environment. The available resources in a cloud environment include CPU, RAM, storage, and communication links and network equipment such as routers and switches, etc. However, the percentage of each resource used by different service units is significantly different. Therefore, service units can be classified as computation-intensive, data-intensive or communication-intensive based on the major resources consumed. Based on the major type service units in a cloud application, we categorise cloud applications into three types: computation-intensive, data-intensive, and communication-intensive.

**Application Workload**: The workload of a cloud application is a set of tasks submitted to the cloud and processed by the application. Every service unit of a cloud application can be considered as a queue that holds incoming tasks. Based on the type of service unit that a task requires, we can categorise the tasks of a cloud application’s workloads into three types: computation-intensive, data-intensive and communication-intensive. As discussed in Section 5.4, the application workload will greatly impact the system performance and energy consumption of cloud applications. Thus, the application workload must be specified in an energy efficiency analysis scenario. In addition, the parameters of each task, i.e. the number
of processes for the task $PT$, the size of data to be processed $DT$, and the type of operation to be required $OT$ should also be specified as they have significant impact on cloud applications’ energy efficiency, as introduced in Chapter 5.

**Cloud System Architecture**: As discussed in Chapter 5, the resources consumed by a cloud application and their system configurations can impact the application’s energy efficiency. For example, as discussed in Section 5.4.1.1, when we increased the number of CPU cores allocated to a CPU-intensive task from one to three, the system throughput increased and average energy consumption decreased. Thus, these factors should also be specified in an energy efficiency analysis scenario. The available resources and their relationships capture the architectural aspects of a cloud system.

**Deployment Strategy**: Different deployment strategies also result in different energy efficiency of cloud applications. For example, as discussed in Section 5.4.2.1, we deployed the file related data-intensive cloud application on one XLarge VM, two Medium VMs and four Small VMs, respectively. When the total amount of data transferred by the cloud application was set to 64GB, the energy consumption increased and throughput decreased when we changed the application deployment strategy from one XLarge VM to four Small VMs. Thus, how to deploy the service units of a cloud application on VMs should be specified in an energy efficiency analysis scenario.

**Signature**: An energy efficiency signature describes a set of invariants that, when matched by elements in a specified model of a cloud application, indicates the likely system performance and energy consumption to the given energy efficiency analysis scenario. The invariants are usually attributes of a cloud application’s workload, a cloud system’s architecture or the application deployment strategy.

In order to support formalisation of such energy efficiency signatures, we have developed a cloud application workload model and a cloud system architecture model. These two models capture the details of the cloud application workload and the cloud system architecture in the energy efficiency analysis scenarios. By mapping the cloud application workload to the cloud system architecture elements, the performance engineer or system architects can specify the deployment strategy.
of the cloud application. The details of the model design are presented in Section 6.2.1 and Section 6.2.2, respectively.

### 6.2.1 Cloud Application Workload Model

Various cloud application workloads need to be specified during energy efficiency analysis process. The collection of accurate system performance and energy consumption data of cloud applications relies on the use of load test based on a realistic user behaviour model and running the load test in a real-world cloud environment. In order to conduct a reliable evaluation, a realistic and comprehensive cloud application workload model is required. The workload of a cloud application is a set of tasks that application users submit to the cloud application. A cloud application consists of several functional service units. Every service unit of a cloud application can be considered as a queue that holds incoming tasks. Tasks are not required to follow the same processing route. In other words, different tasks submitted by different application users may have different process as users’ behaviours are different.

For example, two online shopping Web site clients A and B submit two tasks to query inventory at the beginning. However, after inventory query, client A may submit a task to make an order while client B may submit a task to register a new account. Even for the same client A, after querying inventory, he/she may choose to make an order or logout with different probabilities. Thus, we need a workload modelling technique so that both cloud application user behaviours and application data can be modelled.

A lot of research effort has been devoted to modelling the workload of cloud applications [48, 50, 53, 55, 117, 118]. However, most existing approaches provide only a fairly basic model for user behaviour: a sequence of user requests on cloud servers arranged into repeating groups with multiple threads (to mimic large number of cloud users) [48, 117]. In order to overcome abovementioned limitation, we have adopted the *stochastic form chart* modelling technique, a widely used approach which can capture realistic user behaviours of web applications [119, 120], to model the cloud application workload [91]. A form chart model is a technology independent bipartite state diagram used to simulate user behaviours of
submit/response systems [89]. It describes what the user sees as system output, and what the user provides as system input at a high level. It captures the structure of the target system from the user’s perspectives and can be augmented with probabilities to capture user interactions with the target system.

A stochastic form chart model is extended from a basic form chart model with stochastic functions to generate performance testing workloads of Web applications [90]. In addition to the form chart, a form-oriented model specifies message types and user data for all the pages and actions. Stochastic form chart models can be extended to model cloud application workloads. Cloud application users send service requests to the cloud environment and get responses from the cloud with service results. Therefore, a cloud system can be considered as a submission/response system. In our approach, cloud application requesting service behaviours are modelled using stochastic form charts.

![Figure 6.1 Simplified Stochastic Form Chart Meta-Model](image)

Figure 6.1 shows a simplified stochastic form chart meta-model. The meta-model consists of abstractions Action, Page and Transition. Abstraction Action represents cloud user requests information, including type of requests, type of resources demanded, data passed to cloud system and responses required. Abstraction Pages represent possible states of the cloud application. Abstraction Transition represents association between actions and pages. A stochastic form chart model is an instance model of the stochastic form chart meta-model and captures the structure of the target cloud application from users’ perspective. It captures user and client service behavioural interaction with the target cloud application services.
In our approach, a cloud application workload model is an extended stochastic form chart model which includes a set of tasks and transition probabilities between tasks modelling the cloud application’s user behaviour. It also models the target cloud application’s behaviour by specifying the service units that each task requires. As introduced in Chapter 4, service units can be classified as computation-intensive, data-intensive or communication-intensive based on the major resource consumed in cloud environment. Based on the type of service unit that a task requires, we can categorise the tasks of a cloud application into three types: computation-intensive, data-intensive and communication-intensive. In real applications, a cloud application task is usually made up of composite tasks that may require multiple types of service units [100]. Thus, we introduce a “composite task” in our workload model to represent such composites.

For each task, a stochastic form chart is created to specify the detailed user requests and required responses from the cloud system. Various properties of the user requests and required responses are captured, such as request type, request delay time, response data and response timeout. For each type of task, the task parameters specified in our energy consumption model, i.e. the number of processes for the task $PT$, the size of data to be processed $DT$, and the type of operation to be required $OT$, are also captured in the stochastic form chart. The high-level class diagram of our cloud application workload model is shown in Figure 6.2.
Figure 6.2 High-level Class Diagram of Cloud Application Workload Mode
6.2.2 Cloud System Architecture Model

Choice of different cloud architectures will usually result in different performance and energy consumption of the overall cloud application as discussed in Section 5.4. Thus, different arrangements of cloud system architectures should be applied during the energy efficiency evaluation process. In order to capture the characteristics of cloud systems, we developed a general abstract cloud system architecture model.

A cloud system architecture model captures the available resources, their relationships and detailed configurations in cloud environments. The available resources in a cloud system include physical servers, VMs, RAM, storage etc. Each available resource is represented by an element in the architecture model. The connections between the elements represent the relationship between different resources. The attributes of each element represent its detailed configurations. For instance, a physical server in a cloud system architecture model has attributes such as name, location, number of VMs available etc. This cloud system architecture model is inspired by the experiments design and results analysis described in Chapter 5. In order to achieve general applicability, we take the general and essential architectural elements in most cloud systems into account in our cloud system architecture model.

![Class Diagram of Cloud System Architecture Model](image)

Figure 6.3 Class Diagram of Cloud System Architecture Model
Figure 6.3 presents the class diagram of the cloud system architecture model in our approach. By adding/removing the elements and changing the value of the attributes, system configuration can be changed accordingly. After mapping the tasks defined in the workload model to the VMs specified in the cloud system architecture model, performance engineers or system architects can decide the deployment strategy of the given cloud application.

6.3 Energy Efficiency Patterns and Signature Specification

Each energy efficiency pattern has a corresponding signature which indicates the values of some attributes specified in the application workload model and cloud system architecture model. We do not attempt to deliver a complete list of energy efficiency signatures for all cloud applications. However, we try and cover the energy efficiency signatures derived from the experiment results presented in Chapter 5. These signatures are not related to any specific hardware type. Therefore, general applicability of these energy efficiency signatures can be achieved. Specific hardware types could be added to signatures to produce more precise estimates of energy consumption.

We use OCL, a declarative and formal language based on first order logic and set theory, to describe these signatures. Formalising these signatures helps energy efficiency analysis tools to automate or semi automate the energy efficiency analysis process. If a pattern describes the scenario in which the energy consumption of the cloud application is efficient, then the pattern is defined as an “energy efficiency pattern”. On the contrary, if a pattern describes the scenario in which the energy consumption of the cloud application is inefficient, then the pattern is defined as an “energy efficiency anti-pattern”. These formalised signatures either specify an energy efficiency pattern or an anti-pattern.

In this section, we present the energy efficiency patterns/anti-patterns and signature specifications developed from our Chapter 5 experiment results.

**Resource-Overcommitted Anti-Pattern**: On an over-committed physical server in cloud environments, aggregated capacity requested by the provisioned VMs exceeds the actual physical capacity of the physical server. As presented in
Section 5.4.1.1 test set 5.4.1.1.5, we deployed CPU-intensive service units on two Medium VMs, two Large VMs and two XLarge VMs, respectively. When we changed the resource allocation strategy from non-overcommitted case (two Medium VMs) to overcommitted case (two Large VMs or two XLarge VMs), the energy consumption of each task increased and system throughput decreased as presented in Figure 5.11 and Figure 5.12, respectively. Thus, if the aggregate resource usage exceeds the available physical capacity of the shared resources, then there may be two impacts on the system: 1) service response time of some or all requests may significantly degrade (performance issue); and 2) energy consumption of the system may increase (energy consumption issue).

In addition, a resource overcommitted allocation strategy may lead to some VMs or even the whole cloud system crashing [121]. Thus, resource over-commitment should be avoided at any time to improve energy efficiency and stability of cloud systems, especially for CPU-intensive workload. An energy efficiency pattern related to resource over-commitment can be summarised as the following signature (informally expressed): For all VMs which are alive and running CPU-intensive tasks, the total number of virtual CPUs should be less than the total number of available physical CPUs on corresponding physical server. The detailed description and OCL-specified signature of this anti-pattern is presented in Table 6.1.

**Table 6.1 OCL-specified signature of Resource-Overcommitted Anti-Pattern**

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
</tr>
</thead>
</table>
| 1  | context CloudSystem inv Resource-Overcommitted:  
|    | self.PhysicalServer -> forall( p : PhysicalServer |  
|    | ( p.VirtualMachine -> select(v : VirtualMachine |  
|    | v.isPowerOn = true  
|    | and v.ServiceUnit -> exist( s : ServiceUnit |  
|    | s.type = 'CPU-intensive' ) )  
|    | -> collect(v: VirtualMachine |  
|    | v.numberOfVirtualCores )  
|    | -> sum() ) <= p.numberOfPhysicalCores ) |

For a given Physical Server, all VMs which are alive and running CPU-intensive tasks, the total number of virtual CPUs should be less than the total number of available physical CPUs on the given Physical Server.
**Hyper-Threading Anti-Pattern**: Hyper-Threading technology allows a single physical processor core to behave like two logical processors, essentially allowing two independent threads to run simultaneously. Hyper-Threading lets multi-threaded software’s threads execute in parallel on a single core, thereby improving system performance. However, Hyper-Threading causes extra power consumption of the physical server. In addition, the extra system throughput obtained by enabling hyper-threading is not as much as expected. Thus, there is a significant increase in the energy consumption of each task as presented in Section 5.4.1.1 test set 5.4.1.1.2. This suggests that Hyper-Threading be disabled for CPU-intensive tasks since there is little to gain from Hyper-Threading if a processor core’s execution resources are already well utilised. Thus, an energy efficiency pattern related to Hyper-Threading can be summarised as following: Hyper-Threading of the physical CPUs should be disabled if any of the VMs alive is running CPU-intensive tasks. The detailed description and OCL-specified signature of this anti-pattern is presented in Table 6.2.

**Table 6.2 OCL-specified signature of Hyper-Threading Anti-Pattern**

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>self.PhysicalServer -&gt; select ( p1 : PhysicalServer</td>
</tr>
</tbody>
</table>

For a given Physical Server, Hyper-Threading of the physical CPUs should be disabled if any of the VMs alive is running CPU-intensive tasks.

**Memory-Usage Anti-Pattern**: Memory is an essential resource for any computing system, and is frequently a performance-limiting factor in cloud environments. Enough memory should be allocated to VMs in order to hold the workload of cloud applications which will run on the VMs, thus minimising virtual memory thrashing. Intuitively, more memory leads to higher system performance, especially for memory-intensive cloud applications.
However, as presented in Section 5.4.1.2, when we increased the size of memory allocated to the VMs which had memory-intensive tasks running on it, the power consumption of the physical server were almost the same. In this case, the task execution time becomes the major factor which impacts the energy consumption of each task while power consumption of the cloud server remained at the same level. Task execution time can be reduced by allocating more available memory to the VMs. Therefore, we should take full advantage of all available physical memory on the cloud server. In addition, similar to CPU over-commitment, memory over-commitment should be avoided in order to ensure energy efficiency. Thus, an energy efficiency pattern related to memory usage can be summarised as following: For all VMs which are alive and running memory-intensive tasks, the total amount of virtual memory should be equal to the total amount of available physical memory on corresponding physical server. The detailed description and OCL-specified signature of this anti-pattern is presented in Table 6.3.

### Table 6.3 OCL-specified signature of Memory-Usage Anti-Pattern

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
</tr>
</thead>
</table>
| 3  | context CloudSystem inv Memory-Usage:  
  self.PhysicalServer -> forall (p : PhysicalServer | 
  ( p.VirtualMachine -> select(v : VirtualMachine | 
    v.isPowerOn = true 
    and v.typeOfServiceUnit = 'memory-intensive') 
  -> collect(v: VirtualMachine | 
    v.numberOfVirtualMemory) 
  -> sum() ) = p.numberOfPhysicalMemory ) |

For a given Physical Server, all VMs which are alive and running memory-intensive tasks, the total amount of virtual memory should be equal to the total amount of available physical memory.

**Task-Parameter Pattern:** For a specific task, different task parameters result in different system performance and energy consumption of the cloud application. For each task in our energy consumption model, task parameters include the number of processes for the task $PT$, the size of data to be processed $DT$, and the type of operation to be required $OT$. As discussed in Section 5.4.1 test set 5.4.1.1.1, Section 5.4.2.1 test set 5.4.2.1.1, Section 5.4.2.2 test set 5.4.2.2.1 and Section 5.4.3 test set 5.4.3.1, for all three types of tasks, when the size of data to be processed ($DT_{comp}^j$),
\( DT_{data}^i, DT_{comm}^i \) increases, energy consumption of the task increases and system throughput decreases if we fix the system configuration and other task parameters. This is because large data size results in longer execution time. Thus, the system throughput decreases and energy consumption increases.

In addition, as introduced in Section 5.4.2, when the number of processes for a file-related data-intensive task \( PT_{data}^i \) increases, the system throughput decreases and energy consumption increases if we fix the system configuration and other task parameters. Intuitively, more concurrent processes introduce synchronisation overhead, which will result in longer execution time of a task. Thus, the system throughput decreases and energy consumption increases.

In summary, for any two workload models “workload1” and “workload2” of a cloud application, if 1) the number and type of task, and the relationship between tasks of “workload1” and “workload2” are the same; 2) the system configurations and application deployment strategies are the same, then bigger number of processes or bigger size of data to be processed of same task will result in higher energy consumption and lower system throughput.

To simplify the description, we cut down the scope of this pattern to single tasks. Thus, an energy efficiency pattern related to task parameters can be summarised as following: for any two tasks of a cloud application, if 1) the type of the tasks are the same; 2) the operation type of the tasks are the same; and 3) the system configuration and service units required by the two tasks are the same, then the task which has bigger number of processes or bigger size of data to be processed will result in higher energy consumption and lower system throughput. The detailed description and OCL-specified signature of this pattern is presented in Table 6.4.
Table 6.4 OCL-specified signature of Task-Parameter Pattern

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
</tr>
</thead>
</table>
| 4  | **context** ApplicationWorkload **inv** Task-Parameter:  

```plaintext
self.Task -> forAll (t1, t2) | ( t1 <> t2 
and t1.taskType = t2.taskType  
and t1.serviceLocation = t2.serviceLocation  
and t1.typeOfOperation = t2.typeOfOperation  
and t1.numberOfProcesses = t2.numberOfProcesses  
and t1.sizeOfDataProcessed > t2.sizeOfDataProcesses)  
implies ( t1.averageEnergy > t2.averageEnergy  
and t1.averageThroughput < t2.averageThroughput)  
and ( t1 <> t2  
and t1.taskType = t2.taskType  
and t1.serviceLocation = t2.serviceLocation  
and t1.typeOfOperation = t2.typeOfOperation  
and t1.numberOfProcesses > t2.numberOfProcesses  
and t1.sizeOfDataProcessed = t2.sizeOfDataProcesses)  
implies ( t1.averageEnergy > t2.averageEnergy  
and t1.averageThroughput < t2.averageThroughput))
```

For any two tasks of a cloud application, if 1) the type of the tasks are the same; 2) the operation type of the tasks are the same; and 3) the system configuration and service units required by the two tasks are the same, then the task which has bigger number of processes or bigger size of data to be processed will result in higher energy consumption and lower system throughput.

**Task-Composition Anti-Pattern**: Cloud application deployment strategies can significantly impact the energy efficiency of the application. In real-world cloud environments, a cloud application task is usually made up of composite tasks that may require multiple types of service units as introduced in Section 6.2.1. Thus, we introduce a “composite task” in our workload model to represent such composites.

As presented in Section 5.4.4 test set 5.4.4.2, we profiled the energy consumption of a scientific cloud application - Epigenome. The application’s workload was a set of composite tasks. Each composite task was consisted of computation-intensive and data-intensive tasks. We observed that deploying the cloud application on VMs with different types of services units on same VM is more energy efficient than separating different types of service units on different VMs. Deploying different service units required by tasks inside a composite task will introduce more communication overhead between VMs, which will result in more
processing time of each composite task. Thus, an energy efficiency pattern related to
task composition can be summarised as following: For each composite task in the
cloud application workload, the service units required by the single type of task
inside the composite task should be deployed on the same VM. The detailed
description and OCL-specified signature of this anti-pattern is presented in Table 6.5.

Table 6.5 OCL-specified signature of Task-Composition Anti-Pattern

<table>
<thead>
<tr>
<th>ID</th>
<th>Signature</th>
</tr>
</thead>
</table>
| 5  | `context ApplicationWorkload inv Task-Composite:
  self.CompositeTask -> forall (t: CompositeTask |
  t.Task -> forall (t1, t2 |
  t1 <> t2 implies t1.serviceLocation = t2.serviceLocation))` |

For each composite task in the cloud application workload, the service units required
by the single type of task inside the composite task should be deployed on same VM.

6.4 Discussion

To the best of our knowledge, our approach is the first cloud application energy
efficiency analysis approach which uses formalised signatures. OCL is a formal
and simple notation which is used to express constraints within UML models to get
to a complete specification. It is underpinned by mathematical set theory and logic,
but was designed for usability and is easily grasped by anybody familiar with
object-oriented modelling concepts in general [122]. Thus, using OCL provides a
flexible, formal, familiar and extensible specification approach that can capture
signatures of energy efficiency patterns. Thus, use of OCL also allows easing of
signature validation. These patterns are generic which can be applied to different
cloud applications and cloud platforms. These energy consumption patterns of cloud
applications can be adopted by system architects or performance engineers to
analyse the energy efficiency of a target cloud application. They can also be
extended to develop new energy consumption patterns of cloud applications.

The key limitation of these energy efficiency patterns is that they are extracted
from the experiment results which were conducted with a single type of cloud server
and limited number of selected cloud applications. This may impact the general
applicability of these energy consumption patterns. However, all these energy
consumption patterns presented in the chapter are focused on the logical aspects which are the relationships between application workloads, system configurations, and application deployment strategies. They are not related to a specific hardware model or configurations. We plan to further validate these energy consumption patterns by running more representative cloud applications on different types of cloud servers in the future.

These energy consumption patterns can be applied to predict the trend of system performance and energy consumption. The actual value of energy consumption of the target cloud application cannot be predicted. Actual system performance and energy consumption are highly related to the specific value of task parameters and system configurations. As discussed in Chapter 4, a large amount of experiments with different cloud applications and cloud platforms needs to be conducted in order to get the value of the coefficients and find out the shape of the formulas in our energy consumption model. This is due to the heterogeneity of cloud applications and platforms.

6.5 Summary

In this chapter, we introduced the energy consumption patterns that we derived from the experiments results discussed in Chapter 5, which are formalised as “Signature” using OCL. We first presented the energy efficiency pattern definition schema, which includes all the detailed information required in analysing the energy efficiency of a cloud application. Then we introduced the cloud application workload model and cloud system architecture model which are designed to support the formalised “Signatures”. The details of each pattern and its OCL-specified signature were presented. Finally, we discussed the key advantages and limitations of our approach to energy efficiency analysis of cloud applications. The implementation of an energy consumption pattern checker is described in Chapter 7. These energy consumption patterns will be further validated by our case studies presented in Chapter 8.
Chapter 7

StressCloud – Automating Performance and Energy Consumption Analysis

As discussed in Chapter 5, we conducted extensive experiments to collect fine-grained system performance and energy consumption data with varying system configurations and workloads based on different types of tasks. However, manually conducting these experiments and analysing the results was very error-prone and time consuming. In addition, we developed a set of energy efficiency signatures of cloud applications as presented in Chapter 6. However, it is also very tedious to manually apply these signatures to analyse energy efficiency of a given cloud application. A tool which can effectively use these formalised signatures and automate the energy efficiency analysis is needed. In order to address the abovementioned issues, we have developed an automatic performance and energy consumption analysis tool named StressCloud. This chapter presents the detail approaches for StressCloud.

This chapter is organised as follows. Section 7.1 gives a brief introduction to our motivation. Section 7.2 presents the outline of StressCloud. We describe the major features of StressCloud by using JPetStore as an example in Section 7.3. Section 7.4 presents the design and implementation of StressCloud. Section 7.5 describes the evaluation of StressCloud. Section 7.6 discusses the advantages and limitations of StressCloud. Section 7.7 summarises this chapter.

7.1 Introduction

Profiling and analysing system performance and energy consumption in cloud environments are time consuming. Extensive experiments with different parameters, metrics and workloads need to be conducted. Manual generation of load test plans,
deploying applications, change of system configurations and application of load tests are very tedious and error-prone. In addition, most of existing cloud system performance and energy profiling approaches limit the types of tasks running in the profiling process to only discrete individual types [15]. In real-world cloud environments, users send mixtures of computation-intensive, data-intensive and communication-intensive tasks to cloud systems simultaneously. The way that different types of runtime tasks are composed and deployed will impact the performance and energy consumption of the cloud application [4].

However, it is extremely difficult to manually change task composition and apply the workload to the composed tasks. Therefore, finding the best deployment strategy and system configuration to maximise energy efficiency of the cloud application is extremely challenging. In addition, we introduced a set of energy consumption patterns formalised as signatures of cloud applications in Chapter 6. However, it is also very error-prone and time consuming to validate these signatures by manually comparing the values of the invariants specified in the signatures. Another key limitation of the existing energy efficiency analysis approaches is the lack of automated tool support for energy efficiency evaluation using formalised signatures.

In order to address the abovementioned limitations, we have developed StressCloud, an automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments. StressCloud supports modelling of realistic cloud application workloads, automatic generation of load tests, and profiling of system performance and energy consumption. It also supports automatically analysing energy efficiency of cloud applications by using the formalised signatures before running load test. It adopts the cloud application workload model and the cloud system architecture model introduced in Chapter 6. By using StressCloud, system architects and performance engineers can effectively and accurately evaluate the system performance and energy consumption of cloud applications.
7.2 Overview of StressCloud

Our aim is to support the analysis of the trade-off between system performance and energy consumption of cloud applications. We aim to reduce the time and effort of conducting load test and trade-off analysis while guaranteeing the effectiveness and accuracy of the performance and energy evaluation process. StressCloud is expected to assist cloud system architects and performance engineers to find alternatives of cloud application designs and deployments. It is also expected to assist system architects and performance engineers derive both static and dynamic system-level performance optimisation strategies for cloud applications.

In an attempt to hide the complexity of the load test and energy measurement from cloud systems architects and performance engineers, StressCloud is designed to perform load test automatically based on cloud system architecture and cloud application workload models specified by end users. Therefore, system architects and performance engineers can focus on analysing the system performance and energy consumption results. In addition, StressCloud can automatically validate the energy efficiency of a given cloud system architecture and cloud application workload by using formalised signatures. This will significantly reduce the complexity of the load test suites and performance data collection, which in many cases require more time to understand and develop than the testing itself.

In this section, we briefly outline how StressCloud supports the modelling of cloud application workloads, generates and performs realistic cloud load tests, automatically profiles and visualises both system performance and energy consumption data for analysis, and validate the energy efficiency of the cloud application under a given load test scenario. Based on (i) a specified high-level workload model and (ii) a cloud system architecture model specified by the user, StressCloud can automatically generate and deploy load test service units for a cloud system and generates large-scale, realistic load tests. It can also automatically profile the system performance and energy consumption of the cloud system. Before running load tests, StressCloud can try and validate the energy efficiency of a given workload model and cloud architecture model based on our formalised energy usage signatures. For an existing cloud application, the performance and energy
consumption can be obtained using generated load tests. For proposed cloud application or what-if analysis of proposed re-engineering changes to a cloud application, we allow users to model cloud application service workload composed from different types of tasks and service inter-dependencies. Those workload models are then deployed and run on real-world cloud hardware platforms to produce a realistic performance and energy consumption behaviours for measurement and comparison.

**Figure 7.1 StressCloud Performance and Energy Consumption Analysis Process**

Figure 7.1 shows how StressCloud is used to validate a cloud application’s energy efficiency, and perform load tests to profile the performance and energy consumption of a cloud application. First, the system architect or performance engineer defines the cloud application workload model (1 in Figure 7.1). The workload model is composed of a set of tasks modelling the target cloud application behaviour. Based on our energy consumption model presented in Chapter 4, we
categorise runtime tasks into three types: computation-intensive, data-intensive and communication-intensive. In real-world applications, cloud application workloads are made up of composite tasks that consume multiple types of cloud resources, including CPU, RAM, data storage and network devices. Thus, we introduce a “composite task” in our workload model to represent such composite tasks. This model is then further augmented by the system architect or performance engineer with transition probabilities and properties between tasks, forming a detailed, executable workload model.

We have developed a collection of cloud services units configured by these application workload models in order to provide a realistic target application for energy and performance data collection. These services units respond to user requests by performing tasks defined in the workload model. This allows for what-if energy consumption and system performance analysis of planned systems and for modelling the re-engineering of existing systems. For each task, a stochastic form chart is created to specify detailed user requests and required responses from the cloud system.

StressCloud can also be used to stress a deployed existing cloud application. In this case, the system architect or performance engineer specifies which deployed cloud service to invoke, including parameter data sent to the deployed application service. At run-time, real system requests are sent by StressCloud to stress-test the target application.

A cloud system architecture model is then defined by the performance engineer or system architect to specify the elements in the target cloud system (2 in Figure 7.1). Our cloud architecture model includes all available resources in the target cloud system and their detailed configurations. It is extended from the cloud system architecture model introduced in Chapter 6 by 1) adding more architectural elements, e.g. a “ServiceGroup” is added to contain all available service units deployed on a specific virtual machine; and 2) adding more attributes of existing architectural elements in order to specify the deployment options in detail, e.g. an attribute named “frequencyOfPhysicalProcessor” is added to the element “PhysicalServer” to specify the frequency of the physical processors in the related physical server. In the cloud system architecture model, resources of different types
in the cloud can be specified by different resource locations, such as physical servers and virtual machines. After mapping the tasks defined in the user workload model to corresponding service unit deployed on resources in the cloud system architecture model, workload deployment scripts are generated (3 in Figure 7.1).

The energy consumption patterns of cloud applications described in Chapter 6 have been integrated into StressCloud. Before deploying the cloud application and conducting load tests, system architects or performance engineers can validate the specified cloud system architecture model and application workload model against the patterns we have derived (4 in Figure 7.1). The validation results indicate an expectation of the energy efficiency of the given load test scenario. Engineers can modify their application based on this feedback without running expensive and time-consuming empirical tests on the model or target application. Empirical tests are still needed to gather accurate performance predictions but our energy performance patterns reduce the need for these.

Based on the generated deployment scripts, load test service units are uploaded and deployed to the virtual machines in the target cloud system (5 in Figure 7.1). These load test service units were developed based on our previous research that incorporated computation-intensive and data-intensive tasks, and supported service-to-service communication-intensive tasks described in Section 5.4. Load test scripts are then automatically generated based on the workload model (6 in Figure 7.1). Next, the specified load tests are performed automatically in the target deployed cloud system based on the load test scripts (7 in Figure 7.1).

The performance and energy consumption data of the target cloud system are collected (8 in Figure 7.1) and then visualised using a variety of charts (9 in Figure 7.1). The visualised system performance and energy consumption data are updated at a user-specified rate, by default every 20 seconds. The test results are stored for future reference and comparison with new tests running with differing tasks, loads and deployment models. In addition, new energy consumption patterns can be extracted and formalised from new conducted experiments (10 in Figure 7.1). These new derived patterns can be added to the energy efficiency database for future validation.
7.3 StressCloud Example Usage – JPetStore

JPetStore is an online pet store as introduced in Chapter 4. It has the basic functionalities that most online stores would have, such as browsing and searching the product catalogue, choosing items to add to a shopping cart, amending the shopping cart, and ordering the items in the shopping cart. It is a reference Java application that illustrates how various J2EE platform capabilities can be used to realise complex enterprise and cloud applications. This section illustrates the usage of the key functions of StressCloud using JPetStore as an example.

7.3.1 Cloud Application Workload Modelling

In this subsection we describe how a system architect or performance engineer uses StressCloud to model the workload of JPetStore. We extended the workload model introduced in Section 6.2.1 and integrated it into StressCloud. First, a high-level workload model is defined using an extended stochastic form chart. An example is shown in Figure 7.2. In this example, the client (represented by top left icon) is modelled with a set of requests (represented by large icons with sub-request containers) linked together with transitions (represented by grey arrowed lines) with probability annotations. Start (green) and end (red) circle icons define a state-chart-style model for the workload.

This example specifies that the user selects a task (Signin, GetProduct, GetIndex, GetHelp, GetCategory, GetCart, CreateNewAccount, GetProductDetail, or CheckOut) with different probabilities after starting the workload. Multiple workload profiles can be defined for each cloud application, for example, average customer workload profile, admin workload profile, advanced customer workload profile and automated service workload profile.

Each task in the workload model is a call to a service unit in the cloud application. Each such service unit is modelled as either a call to an existing deployed cloud application service unit or one or more “basic” service unit types (data-intensive, computation-intensive or communication-intensive). These basic service units types are used to build a model of a target cloud application’s services.
A complex cloud application is thus built up of service units comprised of a mix of different types of service units and different workloads using these service units.

Each workload task may comprise of sub-tasks that allow us to define in detail what the task does. We again use the probability-based stochastic form chart formalism to model these sub-tasks. Figure 7.3 shows an example of three JPetStore sub-task models, (a) a computation-intensive sub-task (modelling information processing on a server node); (b) a data-intensive sub-task (modelling database or file processing on a data storage server node); and (c) a communication intensive sub-task (modelling client-to-server or service-to-service communications); StressCloud allows the performance engineer to specify a range of information about each sub-task, as shown in the left parts of Figure 7.3.
Figure 7.2  A JPetStore High-level Workload Model in StressCloud
Figure 7.3 Stochastic Form Chart Models of Computation-intensive Task (a), Data-intensive Task (b) and Communication-intensive Task (c)
7.3.2 Cloud Architecture Modelling

After defining the detailed workload of a target cloud application, the system architect or performance engineer must define the deployment platform for running the application. The cloud system architecture model described in Section 6.2.2 has been integrated into StressCloud. An example of such a cloud architecture model is shown in Figure 7.4. A cloud platform comprises of physical server hosts, virtual machines and networks. Some virtual machines are optimised for data- or computation-intensive tasks. Virtual machines have configuration parameters, for example, virtual memory size and number of compute cores, a host operating system, and deployed application software service units such as web server, database server etc. Physical hosts and networks have various characteristics, for example, server type, number of CPU cores, amount of physical memory, virtual machine hypervisor type, bandwidth, etc. All the details and the platform structure are captured in the cloud architecture model in StressCloud. Different cloud architecture models can thus be defined to model different physical servers, virtual machine configurations, arrangements of networks and servers, different application software deployments and configurations, etc.

Figure 7.4 (a) shows a high-level model of the cloud platform architecture for running the JPetStore cloud application in our example loading and energy analysis tests. This example specifies a data centre with one physical host and three virtual machine groups on which different tasks will be hosted. Each virtual machine has a service group containing all available service units deployed on this virtual machine. Each task requires one or more service units based on the task composition. Various configuration parameters of the physical server are shown at the left top of the figure. For example, the configuration parameter named “NumberOfPhysicalProcessor” specifies the available physical cores of the physical server. The parameter named “NumberOfVirtualMachine” specifies the available VMs created on the physical server.

These cloud platform specifications are used to model and configure actual cloud hardware. Figure 7.4 (b) shows how the system architect or performance engineer specifies a particular virtual machine configuration that has been created for the
physical host machine and its hypervisor, in this case a VMWare hypervisor. The system architect or performance engineer may configure available parameters for the selected virtual machine type as shown at the left bottom of the figure. For example, the system architect or performance engineer can change the number of virtual processor configured on the VM by changing the value of parameter “NumberOfVirtualProcessor”.

![An Example Cloud Architecture Modelled in StressCloud](image)

**Figure 7.4** An Example Cloud Architecture Modelled in StressCloud

### 7.3.3 Model Validation

The completeness and correctness of the workload models and the architecture model need to be validated before the generation of the deployment plan and the
load test plan. A series of constraints are defined in StressCloud for validation. The model cannot be used to generate load tests without such errors being corrected.

These constraints can be grouped into three categories:

1. Model completeness validation for mandatory entities in the workload models and architecture model. For example, “Start” node and “End” node in a high level workload model must be created to state the start point and end point of the workload model.

2. Model entity completeness validation for mandatory properties of model entities. For example, a virtual machine’s service location in a cloud architecture model must be specified.

3. Model correctness validation for rules relevant to the logic of models. For example, the transition probabilities of all outgoing transitions of one task in a workload model cannot exceed 1.0.

A problem description will be generated if the validation of a constraint fails. For each problem, StressCloud provides “QuickFix” solutions. Such a simplified JPetStore high-level workload model with link errors is shown in Figure 7.5 (a). As the transition probabilities of the link between task “GetIndex” and task “Signin”, and the link between task “GetIndex” and task “GetCart” are set to 0.8 and 0.7 respectively, the workload model cannot pass validation. The problem descriptions generated are displayed in Figure 7.5 (b). The system architect or performance engineer can select one of the provided solutions or use the default solution to fix the corresponding validation problem. The transition probabilities in the workload model may be simple random values, fixed values, or complex stochastic probability models (normal distribution or Gaussian random distribution). The transition probabilities generated by the default solution are subject to normal distributions.
After defining the detailed workload of a target cloud application and the deployment strategy for running the application, the trend of system performance and energy consumption can be validated. A set of energy consumption patterns has been defined as presented in Chapter 6. The workload model and deployment strategy are validated against these energy consumption patterns which have been
formalised as “Signatures” using Object Constraint Language (OCL). If the values of attributes specified in these two models match the values of the variants specified in one or more signatures, a suggestion related to energy efficiency of the given workload model or/and architecture model is displayed. Each suggestion either indicates 1) recommended values of the two models’ attributes which will result in high system performance and low energy consumption; or 2) predicted trend of system performance and energy consumption change when some values of the two models’ attributes change. Based on the suggestion, the system architects or performance engineer can decide whether to continue running the load test with current configurations or changing the models as suggested.

Figure 7.6 An Example Workload Model and Architecture Model for Energy Efficiency Validation in StressCloud
Figure 7.6 shows an example workload model and architecture model energy efficiency validation. Figure 7.6 (a) displays a simplified JPetStore high-level workload model including four tasks “GetIndex”, “GetCart”, “CheckOut” and “Search”. As shown in Figure 7.6 (b), the four tasks are deployed on two virtual machines. Both tasks “Checkout” and “Search” require computation-intensive service unit. They will run on VMs named “FEI_VM2” and “FEI_VM1”, respectively. The two VMs “FEI_VM1” and “FEI_VM2” are created on the same physical server.

As indicated in Figure 7.6 (c), four physical cores are available on the physical server. Three virtual cores are configured on “FEI_VM2” and two virtual cores are configured on “FEI_VM1”, as show in Figure 7.6 (d) and (e) respectively. In this case, the total number of virtual cores configured on the two virtual machines has exceeded the total available physical cores. In addition, both virtual machines are in the “running” status, which indicates the computation resource on the physical server is over-committed to the virtual machines. Based on the “Resource-Overcommitted” energy consumption anti-pattern introduced in Section 6.3, the workload model and architecture model displayed in Figure 7.6 (a) and (b) will result in low system performance and high energy consumption. If the system architects or performance engineers choose to validate the energy efficiency of these
models, a validation result will be displayed, as shown in Figure 7.7. The result also provides detailed information about which element in the workload model or architecture model appears to cause the energy inefficiency and where the element is. This allows the system architects or performance engineers to locate the model element and make changes to the models and system configurations.

7.3.5 Deployment Plan Generation

![Figure 7.8 Development Scripts generated by StressCloud](image)

The model of the cloud platform is used to generate automated configuration scripts in order to configure the platform for load test runs. The generation of deployment plans and some example deployment scripts are shown in Figure 7.8. In this example,
the system architect or performance engineer has assigned various JPetStore services to different virtual machines on the available deployment platform, as presented in Figure 7.8 (a). Some machines may run on the same host, some on different hosts. Those on different hosts will communicate via the physical network, while those on the same host will communicate via the virtual vSwitch provided by the hypervisor.

The generated scripts, such as the one shown in Figure 7.8 (b), are uploaded to the target physical and virtual machines by StressCloud and executed to configure them. Within the StressCloud environment, the system architect or performance engineer may save his/her deployment model and define a new one, assign services to different servers/virtual machines in the new deployment model, and generate and run new deployment scripts.

### 7.3.6 Load Test Plan Generation

After generating and deploying the cloud platform and the service configuration scripts, the system architect or performance engineer uses StressCloud to generate the load test plans of the modelled JPetStore application. Figure 7.9 shows an example of generated load test scripts and configurations. In this example, the load test model as in Figure 7.9 (a) is used to synthesise a test script that will be run on a client machine as in Figure 7.9 (b). This script models the state machine that describes, for this workload model, the sequence of tasks, transitions between tasks, probabilities that each task will be carried out, iterations of each task, and the workload.

Each task in the workload model is modelled either by a call to a deployed real-world cloud application service, or as in this example, a model of that service as in Figure 7.9 (c). The service model is a set of data-, computation- and communication-intensive sub-tasks. These tasks are themselves organised into a probabilistic model, designed to capture the data, computation and communication intensive aspects in the cloud application service respectively. A service may itself be modelled as a workload model, making use of other services, thereby modelling the service-oriented architecture of many cloud applications.

As shown in Figure 7.9 (c), each task’s sub-model component includes the parameters of the services involved in the task service model. These parameters
specify, for example, the number of iterations for storage, compute, send and receive activities in the services, the amount of data to store, process, send and receive in each activity, the type of activity, such as insert, update, delete, select in database, the number of cores to use in processing, the amount of memory to use in processing, the number and size of rows to return for the select activity, and so on. The sub-task activities and task workload models thus allow for both coarse- and fine-grained modelling of cloud application services.

Figure 7.9 Load Test Scripts
7.3.7 Load Tests Running and Results Visualising

Once load test scripts have been generated they are uploaded to one or more machines acting as “clients” and run. The system architect or performance engineer may ask for many hundreds or even thousands of instances of the “client” to be run simultaneously in multiple threads, permitting large-scale stress tests to be performed. Clients running the load tests can be hosted on the same or different physical machines, depending on machine availability, to enable analysis of network performance and energy consumption under different loads.

Different types and number of workload tests can be run simultaneously. For example, the system architect or performance engineer may choose to run 100 “average customers”, 20 “advanced customers”, 5 “administrators” and 10 “automated processing” workload models to analyse the relative performance of each and the total. The system architect or performance engineer may then choose to run different subsets and numbers of these workloads to measure relative energy and performance impacts, respectively.

The results of load tests are periodically collected and interactive visualisations are provided to the system architect or performance engineer. Figure 7.10 shows an example of the JPetStore deployment models and workload models shown earlier being stress-tested on SwinCloud, a private cloud that provides a common computational infrastructure to researchers at Swinburne University of Technology. Various virtual machine and physical machine performance metrics can be collected as shown in Figure 7.10 (a). The energy consumption of physical server and router devices are also collected as shown in Figure 7.10 (b). Different collection intervals and parameters can be set. Results can be saved for post-hoc analysis and comparative analysis of different workload and deployment configurations.
7.4 Design and Implementation

Much research has been devoted to building performance testing tools for cloud systems. Some of the existing tools have not considered energy consumption model [48, 49, 123, 124]. Only a few of the existing cloud performance testing tools have taken into account the energy consumption of data centres [55, 57]. However, they are built based on simulation technologies, which is a popular methodology to conduct performance evaluation of new software systems [42, 43]. By modelling the interactions and behaviours of each component of the proposed system based on its architecture, the whole system can be simulated.

Simulation modelling is steadily becoming more practical with the availability of more powerful and inexpensive computing resources. The key limitation of simulation-based modelling is that test results may be inaccurate because of the imperfection of environmental configuration and input data in the simulation [125]. In contrast to simulation, model-based performance test-bed generation [44] provides more accurate test results as a test-bed is close to the real-world software environments. Therefore, a performance test-bed for cloud systems is generated instead of a cloud system simulator.
StressCloud is realised as a set of Eclipse IDE plug-ins. The workflow of StressCloud is presented in Figure 7.1 and explained in Section 7.2. The high-level architecture of StressCloud is presented in Figure 7.11 which focuses on the relationships of different software components in StressCloud. A set of editors are used to support diagrammatic modelling of workload and deployment platform (1 in Figure 7.11). Three key diagram types are used, i.e., the high-level workload model of the cloud application, low-level workload model of each task and the cloud platform architecture model (2 in Figure 7.11). Diagrammatic editors are instantiated for editing these models using the Eclipse Graphical Modelling Framework. The “DiagramRe-arrangeManager” component uses KIELER\(^\text{15}\), a framework which enables automatic layout in all graphical components of the diagrams within the modelling environment, to arrange the graphic model entities into different layouts based on user selection. The workload models are used to synthesize detailed load test scripts by the “LoadTestScriptsTranslator” (3 in Figure 7.11).

\(^{15}\text{https://www.informatik.uni-kiel.de/rtsys/kieler/}\)
The deployment model is used to synthesise configuration scripts for deploying, configuring and instantiating virtual machines, hosted applications and services by the “DeploymentScriptsTranslator” (4 in Figure 7.11). A set of objects are created to traverse the cloud architecture and workload models to generate these scripts. The deployment scripts are sent to a “DeploymentEngine” (5 in Figure 7.11). This component creates and instantiates virtual machines, runs installation scripts on the machines, distributes workload configurations to machines, and configures our workload services hosted on the machines.

The engine is implemented as an agent that interacts with the target cloud system manager. StressCloud currently supports VMware as the target cloud platform hypervisor. Creating templates of deployment scripts to support more platforms is part of our future work. Similarly, a “LoadTestExecutionEngine” invokes the load tests on the deployed cloud application (6 in Figure 7.11). This component also manages database connections, captures real time performance and energy consumption data from the target cloud system, and transforms the collected data into suitable format for visualisation (7 in Figure 7.11).

The “VisualisationEngine”, implemented using Eclipse SWT (8 in Figure 7.11), queries the stored performance and energy data to provide periodic update. All models are saved in the model library as XML files. The “DiagramGenerationManager” component can automatically retrieve the existing models and generate model entities, associations and their visual icons (9 in Figure 7.11). Before deploying the load test scripts and running load test, “SignatureValidator” validates whether the values of some attributes specified in the two models match the values of invariants specified in the signatures, when a “validating energy consumption” option is chosen (10 in Figure 7.11). If there is a match, a suggestion with predicted energy consumption trend is displayed. All available signatures are saved in “EnergyConsumptionSignatureLibrary” as a set of OCL constraints.
7.5 Evaluation

7.5.1 Experimental Validation

In order to investigate the utility of StressCloud, we conducted a set of experiments to analyse the system performance and energy consumption of modelled cloud applications. We adopted the experiments set up introduced in Section 5.2. We first selected modelled JPetStore named MJPetStore as presented in Section 7.3. We evaluated the system performance and energy consumption with different application workloads and system configurations. We adopted the workload introduced in Section 7.3.1 in the experiments. We firstly deployed the cloud application on one Large VM. Then we deployed the cloud application on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on different VMs respectively, named “3Small(D)”. Finally, we deployed the cloud application on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on same VMs respectively, named “3Small(S)”. The application workload was evenly distributed on all three VMs with “3Small(S)”. The initial number of users was set to 10 and the concurrent user requests of each user were set to 100 in this set of tests. Although the total resources such as CPU and RAM allocated were the same, the energy consumption decreased when deploying the cloud application on multiple VMs compared to deploying the cloud application on single VM. The system throughput increased accordingly. When deploying the cloud application on multiple VMs, the service requests of the cloud application were processed in more concurrent processes, which reduced the execution time of the cloud application. We found that better energy efficiency and system performance were achieved when MJPetStore was deployed on three virtual machines and different types of service units on different virtual machines.

We also modelled and evaluated the energy consumption and system performance of a cloud application named Epigenome\(^6\) in StressCloud. Epigenome is a scientific workflow application which maps short DNA segments collected using high-throughput gene sequencing machines to a previously constructed

\(^6\) http://epigenome.usc.edu/
reference genome. The workflow reads DNA files, splits input segment files into small chunks, processes the chunks and merges the mapped genome sequences into a single output map. A data-intensive task was used to model file reading and a computation-intensive task was used to model small DNA chunks processing. We named the modelled Epigenome as MEpigenome. We conducted experiments with different workload models and deployment configurations. We found that deploying MEpigenome on two Medium VMs with all kinds of service units on the same VM is the most energy efficient while achieving the best system performance.

We compared the system performance and energy consumption results of JPetStore and MEpigenome against the existing JPetStore and Epigenome which are presented in Section 5.4. With the same workload models and cloud architecture models, we observed consistent energy consumption and system performance variations of MJPetStore and JPetStore. Similar observations were found between MEpigenome and Epigenome.

In order to further investigate the utility of StressCloud, we conducted three case studies to evaluate the system performance and energy consumption of the selected cloud applications. Detailed experiment results will be presented in Chapter 8.

7.5.2 User Evaluation

We carried out an informal user evaluation of StressCloud within our research group. 7 participants were recruited to try StressCloud. The selected participants had different backgrounds and experience levels which are listed in Table 7.1: 3 participants were postdoctoral research fellows and the other 4 participants were PhD candidates; 2 participants had 2~7 years work experience in industry related to software engineering; 2 participants had experiences in performance engineering. All participants had a rough understanding of cloud computing.
Each participant was asked to profile the energy consumption and system performance of a simple modelled cloud application on one virtual machine. The selected cloud application in this user evaluation contained one computation-intensive task, one data-intensive task and one communication-intensive task. Before starting the experiment, each participant was given a 20 minutes introduction to StressCloud. Then the participants were asked to complete the following tasks by: 1) creating the workload model and augmenting it by specifying the properties of each workload model component; 2) creating the cloud architecture model and augmenting it by specifying the properties of each architecture model component; 3) deploying workload to target cloud platform; 4) generating load test scripts; and 5) running load test and viewing visualised data.

We then interviewed them to gain their perceptions of StressCloud’s utility and usability. The following questions were given to the participants:

1) Do you think StressCloud is useful for automating the energy and performance profiling process of cloud applications?
2) Do you think StressCloud is easy to use?
3) Do you think it is easy to learn how to use StressCloud?
4) Would you like to use StressCloud for automating energy efficiency evaluation for cloud applications in your future projects?
5) Do you have any further comments regarding to the automation process of StressCloud?

---

**Table 7.1 Background and Experience Level of Participants**

<table>
<thead>
<tr>
<th>Participant NO.</th>
<th>Work Experience</th>
<th>Position</th>
<th>Technical Background</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Post Doc</td>
<td>PhD</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7 years</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>2 years</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
All participants found that StressCloud was very useful for automating the energy and performance profiling process of cloud applications. They also found StressCloud was easy to use and understand. Qualitatively, they stated that it was easy to use StressCloud to model the application workload and cloud architecture, and apply load tests. Particularly, 2 participants who had experiences in performance engineering and used JMeter stated that StressCloud was much easier to use and more efficient than JMeter in terms of test scripts generation. StressCloud provides a visualised user interface for the system architect or performance engineer to automatically generate load test scripts while in JMeter all test scripts are manually specified. They also stated that they would like to use StressCloud in their future projects. However, most of them would like a better approach to automate augmenting the workload model and architecture model instead of manually specifying the properties of each model element.

We compared the time taken to evaluate the system performance and energy consumption of JPetStore manually in contrast to using StressCloud. This work efficiency comparison compares the total effort to build workload and energy test-bed models, conduct all the load tests, and profile the system performance and energy data. All of the load tests were undertaken by an experienced system architect or performance engineer who fully understood the target cloud application and the evaluation process of system performance and energy consumption before StressCloud development.

Manually completing the whole process took the author 2 months of full-time work. Repeating this with StressCloud only took 1.5 weeks by the author. An additional one-off overhead of approximately 2 hours was incurred for the engineer to become familiar with StressCloud. However, extensive further analysis activities with StressCloud can be performed within hours or even minutes, i.e. changing deployment models, workload probabilities and tasks, re-running tests. Conducting these analysis activities manually would take extensive application and test redevelopment running into weeks.
7.5.3 Threats to Validity

Here we discuss the threats to the validity of the patterns that we derived from the experimental results:

_Threats to construct validity._ The main threat to the construct validity of our experimental evaluation is the extraction of workload models. A realistic workload model is essential to reflect the user behaviours interacting with the cloud application. Thus, the main threat to construct validity is that whether the workload model can properly demonstrate the utility of StressCloud in helping find the correct deployment configuration to maximise energy efficiency of a cloud application. To minimise this threat, we adopted the JPetStore workload model presented in [126]. This workload model was synthesised from the existing JPetStore based on empirical measurements of user behaviours. By doing so, we could demonstrate the effectiveness of StressCloud in finding an energy efficient deployment configuration of the cloud application.

_Threats to external validity._ The main threat to the external validity of our experimental evaluation is the representativeness of the cloud environment selected in the experiments. We conducted the experiments on a HP Z400 server in SwinCloud. The model of a cloud server in real-world cloud environments can be very diverse. This would impact the power consumption of the server and the load test execution time. Thus, the energy consumption and system performance of the cloud application could be different from our experimental results in terms of actual figures. However, as the resources consumed by the modelled cloud application remain the same if we keep the workload model and deployment configuration the same, the patterns of energy consumption and system performance of the modelled cloud application would still be similar to our experimental results in general. Another threat to the external validity of our experimental evaluation is the representativeness of the target application selected in the experiments. We selected a small e-commerce website JPetStore and a scientific application Epigenome in our experiments. The type of application in real-world cloud environments can be very diverse which will result in different resource consumption. Thus, we selected two
more cloud applications and perform further validation of StressCloud. The experimental results will be presented next in Chapter 8.

*Threads to internal validity.* The main threat to the internal validity of our experimental evaluation is the comprehensiveness of our experiments. Two major factors have been explored to investigate the energy consumption and system performance of cloud applications, including the workload and the deployment configuration. There are two aspects regarding to the deployment configuration – the resource allocation strategy of the cloud application and the hardware component settings of the selected cloud platform. In our experiments, we changed resource allocation strategy of the cloud application and kept hardware component settings of the cloud platform unchanged.

From cloud providers’ perspective, they tend to change resource allocation strategies rather than modifying the hardware component settings when new tasks arrive. Modifying the hardware component settings would potentially result in service failures, which would not be acceptable by both cloud users and providers according to SLAs [127]. Therefore, we believe that we need not further change the hardware component settings in our experiments to evaluate the impact on energy consumption and system performance.

### 7.6 Discussion

We evaluated StressCloud’s effectiveness for supporting automated cloud application workload modelling and load test generation. We reproduced the load tests previously done manually and reported in Chapter 5 using StressCloud. The following actions are conducted to energy and performance test each target cloud application: 1) modelling the client workload of the target cloud application; 2) augmenting the client workload model with probabilities and parameters; 3) modelling the cloud architecture to define the deployment platform; 4) generating the load test and deployment scripts; 5) running the tests to collect system performance and energy data; and 6) comparing new results against those previously obtained manually.
StressCloud allows system architects and performance engineers to compare the performance and energy consumption of a modelled cloud application against a non-modelled existing cloud application, e.g. MJPetStore vs JPetStore. This comparison of performance and energy consumption can be used to guide the abstraction process from a legacy system to a cloud application model in StressCloud. More specifically, the comparison of performance and energy indicates (1) whether the abstraction/refinement has influenced the performance and energy consumption; and (2) whether the performance and energy consumption differ between different models. The comparison of performance and energy consumption also allows engineer to investigate which part of the system causes the difference. We are developing a mechanism to support refining legacy web systems to StressCloud, and will use such performance and energy consumption comparisons to guide users in the abstraction and refinement processes.

Most current cloud application load test tools require significant programming effort of system architects and performance engineers in building prototypical cloud-side service components [50, 51]. This requires considerable knowledge of the cloud application under test and a lot of effort to build and maintain the prototypes, especially for large cloud applications or cloud applications which the system architect or performance engineer is unfamiliar with. In contrast, StressCloud provides the ability to build a model of the cloud application under test by comprising a mix of data, computation and communication tasks under different workloads. In addition, StressCloud can also be used to stress a deployed real-world cloud application. In this case, the system architect or performance engineer only needs to specify which deployed cloud services to invoke, including parameter data to send to the application services.

The key advantages of StressCloud include: 1) its use of a formal model for user behaviour modelling; 2) the ability to automatically generate the load test scripts and cloud service deployment scripts; and 3) the ability to automatically run load test, and profile system performance and energy consumption data. Unlike simulation-based modelling for such analysis [50], StressCloud allows cloud services to be deployed and run in a real-world cloud environment, resulting in more accurate and realistic estimation of cloud application performance and energy
consumption. The stochastic form chart model used by StressCloud allows the system architect or performance engineer to model user behaviours as probabilistic interactions between user request and cloud server response. The accuracy of using stochastic form chart to model user behaviour has been presented and discussed in [90]. System architects and performance engineers can build different versions of workload models for all or part of a cloud application to compare and contrast performance and energy consumption under different user behaviours and deployment configurations.

The major current limitation of StressCloud is the adequacy of the stochastic form chart model to capture the “realistic” user behaviours of cloud applications. StressCloud’s stochastic form charts rely on the system architect or performance engineer specifying user behaviours and probabilities of different user interactions with cloud applications and parameters of invoked cloud services. These are thus as sensitive to erroneous data as in any other performance testing tool. One mitigating approach which we have adopted for StressCloud is to allow multiple form chart models to be defined for the same target cloud application. This allows the system architect or performance engineer to experiment with a variety of different workload modes and compare cloud application’s performance and energy consumption under different conditions.

Augmenting the workload models with probabilistic information and other application parameters related to user behaviours can be a complex process for large-scale cloud applications. Our experiences indicate that a set of wizards could be created for automatic augmentation of workload models. This will still allow system architect or performance engineer to change the data via the question/answer wizards to specify alternative workload models of the cloud application, which is more easily than modifying many variables as at present.

Currently, StressCloud only supports VMware as the cloud platform hypervisor. A key piece of future work is to create a set of templates to traverse the workload and cloud architecture model in order to generate load test scripts and deployment scripts for the selected cloud platform. This will allow a system architect or performance engineer to add more templates to StressCloud to support more cloud platforms.
Finally, we would like to support reverse-engineering of user behaviour models and architecture models in StressCloud in the future. This will allow a system architect or performance engineer to reverse engineer an existing web application behaviour model and architecture model, and then map the workload model and the architecture model to different types of tasks to be run in a real-world cloud environment. In this way, the system architect or performance engineer will be able to analyse the performance and energy consumption of the application after its migration to a cloud environment.

7.7 Summary

In this chapter, we have presented StressCloud, a novel performance and energy consumption analysis tool for cloud applications. StressCloud offers the following features: 1) the ability to model realistic cloud application workloads at varying levels of detail; 2) the ability to model cloud application deployment configurations at varying levels of detail; 3) support for correctness and completeness validation of workload models and cloud application architecture models; 4) the ability to validate energy efficiency of user-defined models based on formalised energy consumption signatures; 5) automatic generation of detailed load test plans; 6) support for automatic load test; and 7) automatic monitoring, profiling and analysis of system performance and energy consumption.

We first gave an overview of StressCloud. Then we introduced the details of each key feature of StressCloud using JPetStore as a realistic usage example. The details of design and implementation of StressCloud were then presented. We presented a set of evaluation results including experimental validation and a user evaluation. Comprehensive experimental validation results of our case studies using StressCloud will be presented next in Chapter 8. Finally, we discussed the key advantages and limitations of StressCloud.
Chapter 8

Case Studies

Throughout this thesis, we have explained in detail how we analyse the system performance and energy consumption of various cloud applications. A set of formalised cloud application energy consumption patterns have been developed, as presented in Chapter 6. An automatic performance and energy consumption analysis tool named StressCloud has been developed to automate load tests and data analysis in the performance and energy consumption evaluation process, as presented in Chapter 7.

In this chapter, we introduce three case studies to 1) further validate the formalised cloud application energy consumption patterns; 2) and demonstrate how we can use StressCloud to further analyse the trade-off between system performance and energy consumption of different cloud applications. The three applications we have chosen are “JPetStore”, “Pebble” and “ImageProcessor”, respectively.

This chapter is organised as follows. In Section 8.1, we give an overview of all three case studies. The experimental setup of all three case studies is presented in Section 8.2. In Section 8.3, we introduce the first case study – JPetStore. The second case study – Pebble – is presented in Section 8.4. The third case study – ImageProcessor – is introduced in Section 8.5. We discuss the threats to validity of the experimental results in Section 8.6. Conclusions drawn from the experimental results are discussed in Section 8.7. Finally, we summarise this chapter in Section 8.8.
8.1 Introduction

There are a variety of applications running in cloud environments where each application exhibits diverse utilisation of resources and a broad range of workloads. In addition, such applications usually have multiple distributed computation and data services with inter-service communication. Therefore, a cloud application typically has workload tasks composed of a mixture of communication-intensive, data-intensive and computation-intensive tasks. Different user requests have different mixes of these workload task types.

We have conducted three case studies to further investigate the impact of application workloads and cloud application deployment strategies on the system performance and energy consumption of cloud systems. For all three case studies, we evaluated the energy efficiency of both modelled applications and real-world applications with the same high-level workload models and cloud architecture models. By comparing the energy consumption patterns of modelled applications and corresponding real-world applications, the practical value of StressCloud approach can be validated. For modelled applications, each task in the low-level workload model is a call to one or more service of “basic” unit type(s) (data-intensive, computation-intensive or communication-intensive). For real-world applications, each task in the low-level workload model is a call to an existing deployed cloud application service unit.

The three case studies we utilised to evaluate the energy efficiency of different types of cloud applications are as follows:

- **Case Study NO.1: Online Java Pet Store - JPetStore.**

Most cloud applications require one or multiple Web servers to handle user requests, and one or multiple database servers to process the database queries in response to user requests. JPetStore is selected as a representative of such types of cloud applications. JPetStore has a mix of database, application server, web server and web user interface components, realised using J2EE technologies. Different parts of the application can be hosted on different cloud hardware platforms. This case study aims to evaluate the energy efficiency of such a
on-line transaction-based, retail cloud applications with different workloads and different Web and database server deployment strategies.

- **Case Study NO.2: Online Blogging System – Pebble.**

  Instead of processing application data through database servers, some cloud applications store data as files on hard disk drives. Pebble is selected as a representative application of such type of cloud application. It is a lightweight and open source blogging tool. Blog content is directly stored as XML files on hard disk drives and accessed dynamically without database operations. This case study aims to evaluate the energy efficiency of cloud applications which have large amounts of file I/O operations with different workloads and deployment strategies.

- **Case Study NO.3: Online Image Processing System – ImageProcessor.**

  High performance computing (HPC) applications often require large amounts of computing resources in cloud environments. ImageProcessor is selected as a representative application of cloud-hosted HPC applications. This is a simple image processing tool which can resize images with different user defined parameters. This case study aims to evaluate the energy efficiency of HPC applications with large CPU and memory usage, focusing on different workloads and deployment strategies.

### 8.2 Experimental Setup

We focus on cloud applications running on a single cloud server in this thesis, as discussed in Chapter 5. Thus, we adopted the same experiment setup introduced in Section 5.2 for all three case studies, including the test-bed (Section 5.2.1) and the data profiling method (Section 5.2.2). Our experiments were conducted in SwinCloud. The specifications of the HP Z400 servers deployed in SwinCloud are presented in Table 5.1. The basic types of VMs used in our experiments are Small, Medium, Large and XLarge, respectively. Table 5.2 shows the detailed specifications of the VMs. The power consumption was measured and managed using PowerNode.
Figure 8.1 presents the energy profiling framework used in these experiments. We selected Tomcat 7.0 as the Web server and Microsoft SQL server 2008 as the database server. For all three case studies, all workloads are generated and then sent to SwinCloud using StressCloud on a client PC instead of third party load generation applications such as JMeter.

Each set of tests was repeated ten times to reduce measurement error. Only one aspect was changed in each test set in order to isolate the other impact factors of system performance and energy consumption. We selected the throughput of the system the key performance indicator (KPI). The throughput is defined as the total number of user interactions requested and completed successfully per second for all three cloud applications.

8.3 Case Study 1: JPetStore

The JPetStore has previously been used as a running example in Chapter 7 to demonstrate the usage of StressCloud. It has the basic functionalities of an online store, such as shopping, searching products, checking out, etc. It thus has a mix of predominantly data intensive and network intensive tasks. It requires a Web server to handle user requests and a database server to process the database queries in
response to user requests. We have conducted four major sets of tests to analyse the energy efficiency of JPetStore with different workloads and Web/database server deployment strategies.

This case study is organised as follows. Section 8.3.1 introduces the details of the test cases used in this case study and Section 8.3.2 presents the experimental results, including modelled JPetStore and real-world JPetStore.

### 8.3.1 Test Cases Design

JPetStore uses a Web server to handle the user requests and a database server to process the database queries in response to user requests. Therefore, it has workload tasks composed of a mixture of communication-intensive tasks, database related data-intensive tasks and (limited) computation-intensive tasks. Different Web server and database server deployment strategies may impact the energy efficiency of JPetStore, e.g., deploying Web server and database server on the same VM vs deploying Web server and database server on different VMs.

We designed and conducted four sets of experiments to evaluate the energy efficiency under different workloads and deployment strategies of both modelled JPetStore and real-world JPetStore, named MJPetStore and RJPetStore, respectively. This was to allow us to compare the StressCloud-generated modelled cloud application (MJPetStore) performance to the real-world (RJPetStore) cloud application performance under the same workload and deployment conditions.

As shown in Figure 8.2, the high-level workload model for both MJPetStore and RJPetStore that we selected in this case study is the sub model presented in Section 7.3.1. This workload model specifies that the user selects a task (GetIndex, Signin, GetCart, SearchProduct or CheckOut) with different probabilities after starting the workload. For example, the user first selects a “GetIndex” task to visit the homepage of the website. Then there is a possibility of 0.1 that the user selects “Signin” task, 0.8 that the user selects “SearchProduct”, and 0.1 that the user selects to terminate the application. In this case study, we changed the ratio of different tasks in the model to reflect different aspects of user behaviours.
In test sets 8.3.2.1 and 8.3.2.2 as presented in Section 8.3.2, we applied the workload model shown in Figure 8.2. This workload model captures the user operation of searching product. We first fixed the cloud application deployment strategy while increasing the number of user requests in test set 8.3.2.1. In test set 8.3.2.2, we fixed the number of user requests while changing the application deployment strategy. We then changed the ratio of tasks “GetIndex” and “Signin” to 0.8, and the ratio of tasks “GetIndex” and “SearchProduct” to 0.1 as presented in Figure 8.2. This workload model captures the operation of ordering product. It was then applied in test set 8.3.2.3 and 8.3.2.4 as presented in Section 8.3.2. We first fixed the cloud application deployment strategy while increasing the number of user requests in test set 8.3.2.3. In test set 8.3.2.4, we fixed the number of user requests while changing the application deployment strategy.

8.3.2 Experiment Results

In order to investigate the impact of the number of user requests and the deployment strategy on the system performance and energy consumption of JPetStore, we conducted four sets of tests in this case study, described as follows:

Test set 8.3.2.1: In this test set, we apply the high-level workload model that captures the operation of searching, keep resource allocation and deployment strategy, while changing the number of user requests.

The objective of this set of test was to investigate the impact of different number of user requests on energy efficiency of MPetStore with workload model that captures the operation of searching. The cloud application was deployed on a
Large VM in this set of tests. We increased the user requests per second from 300 to 450, and then 600. The energy consumption and throughput of MJPetStore and RJPetStore are presented in Figure 8.3 and Figure 8.4, respectively. In order to illustrate small differences in throughput data with different number of user requests, the value of system throughput starts from 60 instead of 0 in Figure 8.4. We observed consistent energy consumption and system performance variations for MJPetStore and RJPetStore. When we increased the number of concurrent user requests, the energy consumption of MJPetStore and RJPetStore increased, as shown in Figure 8.3. Accordingly, the throughput of MJPetStore and RJPetStore decreased, as presented in Figure 8.4. More concurrent user requests will introduce more scheduling and synchronising overhead in the cloud application, which will lead to an increase in the processing time of each task. Thus, the energy consumption increased while throughput decreased.

As discussed in Chapter 4, the parameters of a task taken into account in the calculation of energy consumption include: the number of processes for the task $PT$, the size of data to be processed $DT$, and the type of operation it requires $OT$. In this set of experiment, $DT$ and $OT$ were kept constant. We only changed $PT$ in the workload model. When $PT$ increased, the energy consumption increased and throughput decreased. This result validates the energy consumption pattern presented in Table 6.4: for any two workload models “workload1” and “workload2” of a cloud application, if 1) the number and type of tasks, and the relationship between tasks of “workload1” and “workload2” are the same, and 2) the system configurations and application deployment strategies are the same, then a larger number of processes to be processed of the same task will result in higher energy consumption and lower system throughput.
As one may notice that the value of energy consumption of MJPetStore was larger than RJPetStore, and the value of system throughput of MJPetStore was smaller than RJPetStore. For example, when we set the number of user requests to 300, 1) the energy consumption of MJPetStore and RJPetStore were 4746.3J and 870.9J, respectively; 2) and the system throughput of MJPetStore and RJPetStore were 87.4 and 214.2, respectively. The same energy consumption and system performance variations can also be observed in the other two case studies, as presented in Section 8.4 and Section 8.5. This is because the average response time of each user requests for MJPetStore was longer than RJPetStore. As described in Chapter 7, we have developed a collection of mock cloud service units to deploy in our test cloud environment. These services units respond to user requests by performing tasks defined in the workload model. Each user request sent to MJPetStore was treated as a unique user request and processed by the required service unit without any caching mechanism, even if the same web page was requested. This is to make sure that the workload model of the cloud application is as realistic as possible. However, for RJPetStore, an internal caching mechanism is applied to the user requests that required the same web page. Building responses to user requests requires computation and memory resources. However, with repeated user request served from cache, usage of computation and memory resources is spared and they are freed up to handle many more concurrent requests. Even offloading cache entries to hard disk drives when the cache becomes full, the system performance is still generally better than the system performance without caching. Therefore, the average response time of MJPetStore is significantly longer than RJPetStore. The results suggested that we need to modify the MJPetStore workload.
model to include more data-intensive tasks instead of computation-intensive and communication-intensive tasks.

**Test set 8.3.2.2:** In this test set, we apply the high-level workload model that focus on searching, keep the resource allocation strategy and the number of user requests, while changing the deployment strategies.

Traditionally, a cloud user purchases its own computation or storage resources and scales up the resource capacities to improve system performance on demand. However, this may result in additional operational costs and energy consumption. Therefore, it is necessary to investigate energy efficient cloud resource management to avoid purchasing additional resources. The objective of this set of test was to analyse the energy efficiency of different cloud application deployment strategies while the total amount of resources allocated to the cloud application remained unchanged.

In this set of tests, the number of user requests was changed from 300 to 450, and then 600. The experimental results of MJPetStore and RJPetStore are presented as follows:

1. **Experimental results of MJPetStore**

   We firstly deployed MJPetStore on one Large VM, named “1Large”. Then we deployed MJPetStore on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on the same VMs respectively, named “3Small”. The workload was evenly distributed across all three VMs with “3Small”. JPetStore requires a Web server to handle the user requests and a database server to process the database queries in response to user requests. Different capacity of the Web server and database server may impact the system performance and energy consumption. Therefore, we also investigated the energy efficiency with different Web server and database server capacities while the total amount of resources allocated to the Web server and database server remained unchanged. We deployed MJPetStore on one Small VM with computation-intensive service unit and communication-intensive service unit, and one Medium VM with data-intensive service unit, named “1SW_1MD”. After that, we deployed MJPetStore on one Medium VM with computation-intensive service unit and communication-intensive service unit, and one Small VM with data-intensive
service unit, named “1MW_1SD”. The energy consumption and system performance of MJPetStore are presented in Figure 8.5 and Figure 8.6, respectively.

![Figure 8.5 Energy Consumption of MJPetStore with Different Deployment Strategies](image1)

**Figure 8.5 Energy Consumption of MJPetStore with Different Deployment Strategies**

![Figure 8.6 Throughput of MJPetStore with Different Deployment Strategies](image2)

**Figure 8.6 Throughput of MJPetStore with Different Deployment Strategies**

Although the total resources such as CPU and RAM allocated were the same, when we changed the deployment strategy from “1Large” to “3Small”, the energy consumption decreased and the system throughput increased dramatically. For example, when the number of user requests was set to 300, the energy consumption decreased by 37.1% and the system throughput increased by 48.9%. With the workload of the cloud application distributed across multiple VMs, the service requests of the cloud application were processed in more concurrent processes, which reduced the total execution time of the cloud application. Meanwhile, we noticed a lower CPU usage with “3Small” which resulted in a slight decrease of server power consumption compared to “1Large”. This was because that deploying MJPetStore on multiple VMs increased the total capacities of Web servers and database servers, which reduced the scheduling and queuing overhead. Thus, the energy consumption was lower and the system throughput was higher.
When we changed the deployment strategy from “3Small” to “1SW_1MD”, the energy consumption increased and the system throughput decreased remarkably. For example, when the number of user requests was set to 300, the energy consumption increased by 58.9% and the system throughput decreased by 46.4%. With deployment strategy “1SW_1MD”, all user requests were firstly sent to a Small VM with computation-intensive service unit and communication-intensive service unit. If a task required data-intensive service unit, the task would call the data-intensive service unit deployed on the Medium VM to complete the task. Compared to “3Small”, the number of concurrent processes was less than deployment strategy “1SW_1MD”, which increased the execution time of the cloud application. Thus, the energy consumption was higher and the system throughput was lower.

Finally, when we changed the deployment strategy from “1SW_1MD” to “1MW_1SD”, the energy consumption decreased and the system throughput increased. For example, when the number of user requests was set to 300, the energy consumption decreased by 15.3% and the system throughput increased by 59.2%. For both deployment strategies “1SW_1MD” and “1MW_1SD”, all user requests were firstly sent to a VM with computation-intensive service unit and communication-intensive service unit. If a task required data-intensive service unit, the task would call the data-intensive service unit deployed on the other VM to complete the task. As presented in Figure 8.2, all tasks in the high-level workload model required communication-intensive service unit while only three tasks in the high-level workload model required data-intensive service unit. More resources allocated to the VM with communication-intensive service unit reduced the execution time of the cloud application. Thus, the energy consumption decreased and the system throughput increased.

In summary, deploying MJPetStore on three Small VMs was the most energy efficient while achieving the best system performance. In this set of test, we applied the high-level workload model that captures searching behaviours. The major tasks of this workload model were communication-intensive tasks and database related data-intensive tasks. Thus, the major factor which influenced the system throughput was not the computation power. The capacities of Web servers and database servers which determined the total number of user requests handled per second was the key
factor. Comparing to the other three deployment strategies, “3Small” had the most processing capacity of both Web servers and database servers. In addition, deploying MJPetStore on multiple VMs made better use of the available CPU cores, which also improved the system throughput. Thus, deploying MJPetStore on three Small VMs was the most energy efficiency.

2. Experimental results of RJPetStore

We also applied the same set of deployment strategies to RJPetStore. We firstly deployed RJPetStore on one Large VM, named “1Large”. Then we deployed RJPetStore on three Small VMs with Web server and database server on the same VMs, named “3Small”. After that we deployed RJPetStore with Web server on one Small VM, and database server on one Medium VM, named “1SW_1MD”. Finally, we deployed RJPetStore with Web server on one Medium VM, and database server on one Small VM, named “1MW_1SD”. The energy consumption and system performance of both MJPetStore and RJPetStore are presented in Figure 8.7 and Figure 8.8.

![Figure 8.7 Energy Consumption of MJPetStore and RJPetStore with Different Deployment Strategies](image1)

![Figure 8.8 Throughput of MJPetStore and RJPetStore with Different Deployment Strategies](image2)
In this set of test, the number of user requests was set to 300, 450 and 600, respectively. We observed similar energy consumption patterns compared to MJPetStore. As presented in Figure 8.7 and Figure 8.8, deploying RJPetStore on three Small VMs has the lowest energy consumption and the highest throughput. Thus, deployment strategy “3Small” was the most energy efficient. With the workload distributed over multiple VMs, the service requests of the cloud application were processed in more concurrent processes, which made better use of available computational resources such as CPU and memory. This reduced the total execution time of the workload and improved the system throughput. Thus, the energy consumption was lower and the system throughput was higher.

As presented in Figure 8.7 and Figure 8.8, when we deployed MJPetStore with deployment strategies “1SW_1MD” and “1MW_1SD”, the energy consumption increased and system throughput decreased when the number of user requests increased. We also found that when we deployed RJPetStore with deployment strategies “1SW_1MD” and “1MW_1SD”, the energy consumption increased when the number of user requests increased. The system throughput decreased when we change the number of user request from 450 to 600, which was the same as MJPetStore. However, the system throughput increased slightly when we changed the number of user requests from 300 to 450 for RJPetStore. Intuitively, if more user requests were processed in a task, the time spent on initialising each task took a smaller proportion in the task completion time. Thus, the average response time of user request was reduced when we changed the number of user requests from 300 to 450. Thus, the system throughput of RJPetStore increased. However, when we changed the number of user requests from 450 to 600, the benefit of a smaller proportion of task initialisation time was cancelled out by scheduling overhead. Thus, the average response time increased and system throughput decreased.

**Test set 8.3.2.3: In this test set, we apply the high-level workload model that captures ordering, keep resource allocation and deployment strategy, while changing the number of user requests.**

The objective of this set of test was to investigate the impact of different number of user requests on energy efficiency of MJPetStore with workload model that captures the operation of ordering. In this set of test, the number of user requests
was set to 300, 450 and 600, respectively. We changed the ratio of tasks “GetIndex” and “Signin” to 0.8, and the ratio of tasks “GetIndex” and “SearchProduct” to 0.1 for the high-level workload model presented in Figure 8.2. The energy consumption and system performance of MJPetStore and RJPetStore are presented in Figure 8.9 and Figure 8.10. We observed consistent energy consumption and system performance variations of MJPetStore and RJPetStore.

When we increased the number of concurrent user requests, the energy consumption of MJPetStore and RJPetStore increased, as shown in Figure 8.9. Accordingly, the throughput of MJPetStore and RJPetStore decreased, as presented in Figure 8.10. In order to illustrate small differences in throughput data with different number of user requests, the value of system throughput starts from 140 instead of 0 in Figure 8.10. More concurrent user requests will introduce more scheduling and synchronising overhead in the cloud application, which will increase the processing time for each task. Thus, the energy consumption increased and throughput decreased.

In this set of experiment, the size of data to be processed $DT$ and the type of operation it requires $OT$ were kept constant. We only changed the number of processes $PT$ for each task in the workload model. When $PT$ increased, the energy consumption and throughput decreased. This result further validated the energy consumption pattern presented in Table 6.4.

![Energy Consumption with Different Numbers of User Requests](image)

**Figure 8.9 Energy Consumption with Different Numbers of User Requests**
Test set 8.3.2.4: In this test set, we applied the high-level workload model that captures ordering, keep the resource allocation strategy and the number of user requests, while changing the deployment strategies.

As we discussed in test set 8.3.2.2, different cloud application deployment strategies did impact the energy efficiency of the cloud application, even if the total amount of resources allocated to the cloud application remained unchanged. In this set of test, we aimed to verify this finding with a different high-level cloud application workload model. Instead of applying high-level workload model that captures searching, we applied the high-level workload model that captures ordering which was the same workload model as test set 8.3.2.3 in this set of test. The number of user requests was set to 300, 450 and 600, respectively. The experimental results of MJPetStore and RJPetStore are presented as follows:

1. Experimental results of MJPetStore

We adopted the deployment strategies in test set 8.3.2.2 in this set of test. We firstly deployed MJPetStore on one Large VM, named “1Large”. Then we deployed MJPetStore on three Small VMs with computation-intensive service unit, data-intensive service unit and communication-intensive service unit on the same VMs respectively, named “3Small”. The workload was evenly distributed across all three VMs with “3Small”. After that we deployed MJPetStore on one Small VM with computation-intensive service unit and communication-intensive service unit, and one Medium VM with data-intensive service unit, named “1SW_1MD”. Finally, we deployed MJPetStore on one Medium VM with computation-intensive service unit and communication-intensive service unit, and one Small VM with data-intensive service unit, named “1MW_1SD”. The energy consumption and
system performance of MJPetStore are presented in Figure 8.11 and Figure 8.12, respectively.

As presented in Figure 8.12, the system throughput of deployment strategy “1Large” and “3Small” are at the same level, which is higher than the other two deployment strategies “1SW_1MD” and “1MW_1SD”. In addition, the energy consumption of deployment strategy “1Large” and “3Small” are lower than deployment strategies “1SW_1MD” and “1MW_1SD”. With workload model applied that captures ordering, the major database operations performed to process user requests were “insert”. As discussed in Section 5.4.2.2, the “insert” operations read and write large amounts of data on the hard disk drive, which requires more processing power of the VM with data-intensive server unit. Apparently, the total processing power of database server with “1Large” and “3Small” was bigger than “1SW_1MD” and “1MW_1SD”. This resulted in less execution time for each user request. Therefore, the system throughput increased and energy consumption decreased.

![Figure 8.11 Energy Consumption of MJPetStore with Different Deployment Strategies](image)

**Figure 8.11 Energy Consumption of MJPetStore with Different Deployment Strategies**

![Figure 8.12 Throughput of MJPetStore with Different Deployment Strategies](image)

**Figure 8.12 Throughput of MJPetStore with Different Deployment Strategies**
We observed that deployment strategy “1SW_MD” was the most energy inefficient as presented in Figure 8.11 and Figure 8.12. Similar pattern can be found in test results of workload model that captures searching as presented in test set 8.3.2.2. As presented in Figure 8.1, Tomcat was selected as the Web server in our experiments. The maximum number of request-processing threads named “maxThreads” was set to the default value 200. This parameter determines the maximum number of simultaneous requests that can be handled by the Web server. When the number of concurrent user requests exceeds the value of “maxThreads”, the new coming user request will be queued to be processed. This requires a large amount of RAM configured on the VM with Web server deployed. Compared to the other three deployment strategies, the Web server was deployed on one Small VM with “1SW_1MD”, which became the bottleneck when the number of user requests exceeded 200. Therefore, the processing time of each user request increased. Accordingly, the energy consumption increased and system throughput decreased.

2. Experimental results of RJPetStore

Then we applied the same set of deployment strategies to RJPetStore. We firstly deployed RJPetStore on one Large VM, named “1Large”. Then, we deployed RJPetStore on three Small VMs with Web server and database server on the same VMs respectively, named “3Small”. After that we deployed RJPetStore with Web server on one Small VM, and database server on one Medium VM, named “1SW_1MD”. Finally, we deployed RJPetStore with Web server on one Medium VM, and database server on one Small VM, named “1MW_1SD”. The energy consumption and system performance of MJPetStore and RJPetStore are presented in Figure 8.13 and Figure 8.14, respectively.

![Figure 8.13 Energy Consumption of MJPetStore and RJPetStore with Different Deployment Strategies](image)
In this set of tests, the number of user requests was set to 300, 450 and 600, respectively. We observed similar energy consumption and system throughput patterns as MJPetStore. Again, deploying RJPetStore on one large VM was the most energy efficient compared to the other three deployment strategies.

8.4 Case Study 2: Pebble

Social networking websites have become more and more popular these days and many of them have migrated into the cloud, such as Facebook\textsuperscript{17}. Deployed in the cloud, such a social networking website requires massive data storage because of its enormous stockpile of photos and posts, which grows rapidly every day. Instead of storing application data in a relational database, a social networking website usually stores its data as different types of files on hard disk drives. Thus, a large amount of file I/O operations need to be performed to process the application data on a daily basis. Pebble is selected as a representative application of such type of cloud applications. It is a lightweight and open source blogging tool. Its blog content is directly stored as XML files on hard disk drives without any database operations and accessed dynamically. This case study aims to evaluate the energy efficiency of cloud applications which have large amounts of file I/O operations with different workloads and deployment strategies.

This case study is organised as follows. Section 8.4.1 introduces the details of the test cases executed in this case study. Section 8.4.2 presents the experiment results, including modelled Pebble (MPebble) and real-world Pebble (RPebble).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{throughput_graph.png}
\caption{Throughput of MJPetStore and RJPetStore with Different Deployment Strategies}
\end{figure}

\textsuperscript{17} http://www.facebook.com
8.4.1 Test Cases Design

Pebble’s blog content is directly stored as XML files on hard disk drives without any database operations and accessed dynamically. Therefore, it has workload tasks composed of a mixture of communication-intensive tasks and file-related data-intensive tasks.

We designed and conducted three sets of experiments to evaluate the energy efficiency of both MPebble and RPebble under different workloads and deployment strategies. This was to allow us to compare the StressCloud-generated modelled cloud application (MPebble) performance to the real-world (RPebble) cloud application performance under the same test load and deployment conditions. The high-level workload model of both MPebble and RPebble is presented in Figure 8.15. This workload model specifies that the user selects a task (HomePage, Login, BrowseBlogEntry, PublishBlog, WriteComments or ReadComments) with different probabilities after starting the workload. For example, the user first selects a “HomePage” task to visit the homepage of the website. Then the user selects a “Login” task to login. After that, there is a possibility of 0.75 that the user selects a “BrowseBlogEntry” task to view existing blogs, and 0.25 that the user selects a “PublishBlog” task to publish new blogs. Finally, the user may choose a “WriteComments” task or a “ReadComments” task after “BrowseBlogEntry” task or “PublishBlog” task. A “BrowseBlogEntry” task mainly involves reading blog files on hard disk drives while a “PublishBlog” tasks mainly involves writing blog files on hard disk drives. In this case study, we changed the probability of user selecting task “BrowseBlogEntry” and task “PublishBlog” after login task in the model to reflect different aspects of user behaviours.

Figure 8.15 High-level Workload Model for Pebble
We applied five types of high-level workloads in this case study. We changed the probability between task “Login” and task “BrowseBlogEntry” from 0.0 to 1.0 in steps of 0.25 in order to increase the proportion of users’ browsing blog entries operations. The browsing blog entry operations read blog files from hard disk drives. Accordingly, the probability between task “Login” and task “PublishBlog” was changed from 1.0 to 0.0 in steps of 0.25 in order to decrease the proportion of users’ publishing blog operations. The publishing blogs operations write blog files to hard disk drives. The types of workloads are presented in Table 8.1.

<table>
<thead>
<tr>
<th></th>
<th>Browse100</th>
<th>Browse75_Publish25</th>
<th>Browse50_Publish50</th>
<th>Browse25_Publish75</th>
<th>Publish100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability between “Login” and “BrowseBlogEntry”</td>
<td>1.0</td>
<td>0.75</td>
<td>0.5</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>Probability between “Login” and “PublishBlog”</td>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
<td>1.0</td>
</tr>
</tbody>
</table>

In test set 8.4.2.1, we evaluated the energy efficiency of both MPebble and RPebble with different high-level workload models. In test set 8.4.2.2, we analysed the energy efficiency of both MPebble and RPebble with different sizes of data processed for each user requests. We applied different application deployment strategies to analyse the energy efficiency of both MPebble and RPebble in the last test set 8.4.2.3. The experimental results of three sets of tests are presented in Section 8.4.2.

### 8.4.2 Experiment Results

**Test set 8.4.2.1:** In this test set, we keep resource allocation and deployment strategy, while changing the high-level workload model.

Research results show that browsing is the most dominant user behaviour which takes up 92% of total user requests of a social network [128]. However, when there is a big event, the amount of publishing request increases as users tend to share photos or comments with other people. Different amount of browsing requests and
publishing requests may result in different energy consumption and system throughput. Therefore, we would like to investigate the energy efficiency of different high-level workload models with different proportion of browsing and publishing requests. Browsing requests mainly involves file reading operations while publishing requests mainly involves file writing operations. In this set of test, we applied five different high-level workload models presented in Table 8.1.

The cloud application was deployed on XLarge VM in this set of test. The size of published or browsed blog contents was set to 1000K. The energy consumption and system throughput are presented in Figure 8.16 and Figure 8.17, respectively. In general, we observed consistent energy consumption and system performance variations from MPebble and RPebble.

Figure 8.16 shows that the energy consumption of the cloud server increases when the proportion of publishing new blogs operations in the workload increases and the proportion of browsing blog entries operations in the workload model decreased. For example, when the workload model changed from “Browse100” to “Browse75_Publish25”, the energy consumption increased by 20%. The system throughput increased by 33% when the workload model changed from “Browse100” to “Browse75_Publish25”. The level of concurrency at which user requests of multiple types of hard disk drive operations are processed is higher than single types of hard drive disk operations, i.e., read/write versus read. Thus, the total execution time of the workload with mixed types of operations will be less. Accordingly, the system throughput increased as shown in Figure 8.17. On the other hand, the power consumption of the cloud server increased as well when we increased the proportion of the write operations. According to [129], the average power consumed by the write operations on hard disk drives is 20% more than the read operations. The increase in server power consumption was cancelled out by the decrease in workload execution time. As a result, the energy consumption increased.

When we changed the workload model from “Browse100” to “Publish100”, the energy consumption increased and the system throughput decreased dramatically. The level of concurrency at which user requests are processed is the same for the two types of workloads. However, compared to the write operations, the read operations take less time to complete for the same data size. The difference in
execution time between the read and the write operations is the seek time. The seek
time of write operations on a hard disk drive is about 1 millisecond higher than the
read operations. Thus, for the same hard disk drive, the throughput of read
operations is higher than the write operations [130]. In addition, the average power
consumed by the write operations is 20% more than the read operations. Thus, the
energy consumption increased and the system throughput decreased when the
workload changed from “Browse100” to “Publish100”. Similar energy consumption
and system performance patterns can be observed when the workload model changes
from “Browse75_Publish25” to “Browse25_Publish75”.

![Energy Consumption with Different High-Level Workload Models](image1)

**Figure 8.16 Energy Consumption with Different High-Level Workload Models**

![Throughput with Different High-Level Workload Models](image2)

**Figure 8.17 Throughput with Different High-Level Workload Models**

**Test set 8.4.2.2**: In this test set, we keep the high-level workload model, resource
allocation and deployment strategy, while changing the size of data to be
processed in response to each user request.

In this set of test, the objective was to investigate the impact of different size of
data to be processed in response to each user request on energy consumption and
system throughput. The experimental results of MPebble and RPebble are presented
as follows:

1. **Experimental results of MPebble**
The cloud application was deployed on an XLarge VM in this set of tests. The size of published or browsed blog contents was set to 500K, 750K and then 1000K. The energy consumption and system throughput are presented in Figure 8.18 and Figure 8.19, respectively.

Figure 8.18 shows that the energy consumption increased when we gradually increased the size of the data to be processed. As the size of the data increased, more processing time was needed for the server to respond to a user request, which led to longer execution time for the same workload. As a result, the system throughput decreased, as shown in Figure 8.19.

As discussed in Chapter 4, the parameters of a task taken into account in the calculation of energy consumption include the number of processes for the task \( PT \), the size of data to be processed \( DT \), and the type of operation it requires \( OT \). In this set of experiment, \( PT \) and \( OT \) were kept constant. We only changed \( DT \) in the workload model. When \( DT \) increased, the energy consumption increased and throughput decreased. This result validates the energy consumption pattern presented in Table 6.4: for any two workload models “workload1” and “workload2” of a cloud application, if 1) the numbers and types of tasks, and the relationship between the tasks of “workload1” and “workload2” are the same, and 2) the system configurations and application deployment strategies are the same, then the size of data to be processed for the same task will result in higher energy consumption and lower system throughput.
2. Experimental results of RPebble

We then deployed RPebble on an XLarge VM. The size of published or browsed blog contents was also set to 500K, 750K and then 1000K. The energy consumption and system throughput of both MPebble and RPebble are presented in Figure 8.20 and Figure 8.21, respectively. Again, we observe similar energy consumption and system throughput patterns as MPebble.
Test set 8.4.2.3: In this test set, we keep the high-level workload model and resource allocation, while changing the deployment strategy.

As we discussed in test set 8.4.2.1, different high-level workload models resulted in different energy consumption and system performance when we deployed the workload models on one XLarge VM. For the same high-level workload models, different deployment strategy may result in different energy efficiency of the cloud application. The objective of this set of test was to investigate the energy efficiency of the cloud application with different deployment strategies. The experimental results of MPebble and RPebble are presented as follows:

1. Experimental results of MPebble

In this set of test, the size of published or browsed blog contents was set to 750K. We first deployed the cloud application on one XLarge VM. This deployment strategy is named “1XLarge”. Then, we deployed the cloud application on two Medium VMs. This deployment strategy is named “2Medium”. The application workload was evenly distributed across the two VMs. The energy consumption and system throughput are presented in Figure 8.22 and Figure 8.23, respectively.

When the cloud application was deployed on one XLarge VM, we observed that the energy consumption and system throughput patterns were the same as we discussed in test set 8.4.2.1. However, the energy consumption and system throughput patterns were slightly different when we deployed the cloud application on two Medium VMs. For example, when we changed the workload model from “Browse75_Publish25” to “Browse50_Publish50”, the energy consumption increased and system throughput decreased. As we discussed in test set 8.4.2.1, the level of concurrency at which user requests of workload “Browse50_Publish50” are processed is higher than workload “Browse75_Publish25”. The execution time of workload “Browse50_Publish50” should be less than the execution time of workload “Browse50_Publish50”. However, multiple VMs will introduce extra scheduling and synchronisation overhead, which cancel out the workload execution time saved. Thus, the execution time of workload “Browse50_Publish50” was longer than workload “Browse75_Publish25”.

When we changed the deployment strategy from “1XLarge” to “2Medium”, the energy consumption increased in general, as presented in Figure 8.22. The system
throughput of “2Medium” was higher than “1XLarge” when we applied workload model “Browse100”, “Browse75_Publish25”, and “Browse50_Publish50”. With the workload of the cloud application distributed across multiple VMs, the user requests sent to the cloud application were processed in more concurrent processes, which reduced the execution time of the cloud application. Thus, the system throughput was higher.

However, when we applied workload “Browse25_Publish75” and “Publish100”, the system throughput of “2Medium” was lower than “1XLarge”, as presented in Figure 8.23. This is because multiple VMs introduced more scheduling and synchronisation overhead, which resulted in longer workload execution time. Therefore, the throughput of “2Medium” was lower than “1XLarge”.

In summary, deploying workload “Browse100”, “Browse75_Publish25” and “Browse50_Publish50” with “2Medium” deployment strategy was more energy efficient, while deploying workload “Browse25_Publish75” and “Publish100” with “1Xlarge” was more energy efficient.

Figure 8.22 Energy Consumption of MPebble with Different Deployment Strategies

Figure 8.23 Throughput of MPebble with Different Deployment Strategies
2. Experimental results of RPebble

In this set of test, the size of published or browsed blog contents was set to 750K. We first deployed the cloud application on one XLarge VM. This deployment strategy is named “1XLarge”. Then, we deployed the cloud application on two Medium VMs. This deployment strategy is named “2Medium”. The energy consumption and system throughput of both MPebble and RPebble are presented in Figure 8.24 and Figure 8.25, respectively. Again, the energy consumption and system throughput patterns are similar as MPebble.

![Figure 8.24 Energy Consumption of MPebble and RPebble with Different Deployment Strategies](image1)

![Figure 8.25 Throughput of MPebble and RPebble with Different Deployment Strategies](image2)

8.5 Case Study 3: ImageProcessor

Scientific computing often requires the availability of a massive number of computers to perform large-scale experiments. Cloud computing has seen tremendous growth, especially for commercial web applications. The on-demand, pay-as-you-go model introduces a flexible and cost-effective means to access IT
resources. For these reasons, the scientific computing community has shown increasing interest in exploring cloud computing. However, the underlying implementation and performance of cloud systems are very different from traditional supercomputing centres. It is therefore critical to analyse of HPC applications in today’s cloud environments to understand the trade-off between system energy consumption and system performance in the process of migration to the cloud.

Recently, the relative number of HPC applications deployed in cloud environments has grown because of the emergence of data-intensive and computation-intensive fields such as biomedical applications. These biomedical applications usually need to process large amount of images [131]. We selected ImageProcessor as a representative of HPC application which requires large amount of image processing. It is an image processing tool which can resize and reformat images with different user defined parameters. In this case study, we evaluated the energy consumption and system performance of ImageProcessor under different workload and deployment strategies.

This case study is organised as follows. Section 8.5.1 introduces the details of test cases executed in this case study. Section 8.5.2 presents the experiments results, including modelled ImageProcessor (MImageProcessor) and real-world (RImageProcessor).

8.5.1 Test Cases Design

ImageProcessor first reads image files from hard disk drives, resizes the image files, and then writes new images files back to hard disk drives. Therefore, it has workload tasks composed of a mixture of CPU-intensive tasks, memory-intensive tasks and file-related data-intensive tasks.

We designed and conducted three sets of experiments to evaluate the energy efficiency of both MImageProcessor and RImageProcessor under different workloads and deployment strategies. This was to allow us to compare the StressCloud-generated modelled cloud application (MImageProcessor) performance to the real (RImageProcessor) cloud application performance under the same test load and deployment conditions. We modelled the high-level workload of MImageProcessor with a composite task composed of one CPU-intensive task, one
memory-intensive task and one file-related data-intensive task. The composite task covered all activities involved in the image processing procedure. The high-level workload model of ImageProcessor is presented in Figure 8.26. The CPU-intensive task in this high-level workload model calculated Fibonacci sequence based on the largest number of the Fibonacci sequence, which was set to 30 in this high-level model. The memory-intensive task processed a file using memory. It consumed as much memory as possible based on the size of memory allocated to it. In this high-level workload model, the size of file and memory allocated to the memory-intensive task was set to 2MB and 1MB, respectively. The file-related data-intensive task read and wrote the same file to hard drive disk. The size of file for the data-intensive task in this high-level workload model was set to different sizes in different test sets, which will be explained in Section 8.5.2.

In test set 8.5.2.1, we evaluated the energy efficiency of both MImageProcessor and RImageProcessor with different resource allocations. In test set 8.5.2.2, we analysed the energy efficiency of both MImageProcessor and RImageProcessor with different deployment strategies. Specifically, we applied different memory allocation strategies to analyse the energy efficiency of both MImageProcessor and RPIImageProcessor in the last test set 8.5.2.3. The experimental results are presented in Section 8.5.2.

![Figure 8.26 High-level Workload Model for ImageProcessor](image)

**Figure 8.26 High-level Workload Model for ImageProcessor**

### 8.5.2 Experimental Results

In order to investigate the impact of size of the data to be processed and the deployment strategies on the system performance and energy consumption of ImageProcessor, we conducted three sets of tests in this case study. In order to better
illustrate the system throughput, we converted the value of throughput from “the total number of user interactions requested and completed successfully per second” to “the total number of user interactions requested and completed successfully per minute” in this case study. The experimental results are described as follows:

**Test set 8.5.2.1: In this test set, we keep the high-level workload model, while changing the deployment strategy to allow more resources.**

As we discussed in previous section, the workload of ImageProcessor is composed of CPU-intensive tasks, memory-intensive tasks and file-related data-intensive tasks. In this set of test, we aimed to investigate the impact of different amount of memory allocated to the memory-intensive task on the energy efficiency of the ImageProcessor. The experimental results of MImageProcessor and RImageProcessor are presented as follows:

1. **Experimental results of MImageProcessor**

In this set of test, the size of image file to process was fixed at 2000K. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was first deployed on a VM which had three cores and 6GB RAM configured. This deployment strategy is named “3Cores_6GBRAM”. In order to investigate whether more memory allocated to the cloud application could increase the energy efficiency, we kept the number of cores allocated unchanged while increased the memory allocated to the cloud application to 8GB. This deployment strategy is named “3Cores_8GBRAM”. The energy consumption and system throughput are presented in Figure 8.27 and Figure 8.28, respectively.

When we increased the number of user requests per second, the energy consumption increased and system throughput decreased as expected. When we deployed the cloud application with “3Cores_6GBRAM”, the energy consumption was higher than deploying the cloud application with “3Cores_8GBRAM”, as presented in Figure 8.27. The system throughput increased when we changed the cloud deployment strategy from “3Cores_6GBRAM” to “3Cores_8GBRAM”, as shown in Figure 8.28. As discussed in Section 5.4.1.2, a cloud application’s memory usage has only slight impact on the total power consumption of the cloud server. Thus, the server power consumption remained at the same level when we changed
the memory allocated from 6GB to 8GB. On the other hand, more memory allocated to the cloud application resulted in shorter execution time. This was because that more memory allocated to the cloud application reduced the usage of disk as a virtual memory space. Therefore, the energy consumption decreased and the system throughput increased.

![Figure 8.27 Energy Consumption of MImageProcessor with Different Memory Allocation](image1)

**Figure 8.27 Energy Consumption of MImageProcessor with Different Memory Allocation**

![Figure 8.28 Throughput of MImageProcessor with Different Memory Allocation](image2)

**Figure 8.28 Throughput of MImageProcessor with Different Memory Allocation**

We also observed a significant decrease in system throughput under deployment strategy “3Cores_6GBRAM” when we increased the number of user requests from 20 to 30. This was because that a VM with 6GB RAM was not powerful enough to handle so many memory-intensive tasks. Therefore, the system throughput decreased largely.

In summary, deploying MImageProcessor with deployment strategy “3Cores_8GBRAM” was more energy efficient than “3Cores_6GBRAM”.

2. **Experimental results of RImageProcessor**

In this set of tests, we set the size of the image file to process to 2000K. In order to compare the experimental results to MImageProcessor, we applied the same
workload models and deployment strategies as MImageProcessor. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was first deployed on a VM which had three cores and 6 GB RAM configured. This deployment strategy is named “3Cores_6GBRAM”. Then we deployed the cloud application on a VM with three cores and 8 GB RAM. This deployment strategy is named “3Cores_8GBRAM”. The energy consumption and system throughput of both MImageProcessor and RImageProcessor are presented in Figure 8.29 and Figure 8.30, respectively. We observed similar energy consumption and system throughput patterns as MPebble. The experimental results show that deploying RImageProcessor with deployment strategy “3Cores_8GBRAM” was more energy efficient than “Cores_6GBRAM”.

![Figure 8.29 Energy Consumption of RImageProcessor with Different Resource Allocation](image1)

![Figure 8.30 Throughput of RImageProcessor with Different Resource Allocation](image2)

Test set 8.5.2.2: In this test set, we keep the high-level workload model and resource allocation, while changing the deployment strategies.

We aimed to analyse the variations of energy consumption and system performance with deployment strategies that had the same amount of CPU and
memory resources allocated. The experimental results of both MImageProcessor and RImageProcessor are presented as follows:

1. Experimental results of MImageProcessor

In this set of test, the size of image file to process was set to 1000K. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was first deployed on one XLarge VM. This deployment strategy is named “1XLarge”. Then we deployed the cloud application on two Medium VMs. This deployment strategy is named “2Medium”. The application workload was evenly distributed across two Medium VMs. The energy consumption and system throughput are presented in Figure 8.31 and Figure 8.32, respectively.

The energy consumption increased and system throughput decreased under both deployment strategies “1XLarge” and “2Medium” when we increased the number of user requests per second from 10 to 40. When the deployment strategy changed from “1XLarge” to “2Medium”, the system throughput increased. As we discussed in test set 8.3.2.2, with the workload distributed across multiple VMs, the user requests were processed in more concurrent processes, which reduced the execution time of the cloud application. Thus, the system throughput increased. However, the energy consumption under both deployment strategies “1XLarge” and “2Medium” were at the same level. For example, when we set the number of user requests per second to 30, the energy consumption of “1XLarge” was 2162.6J and the energy consumption of “2Medium” was 2181.9J. This was because the server power consumption under “2Medium” was higher than “1XLarge”. Multiple VMs introduced higher scheduling overhead which resulted in higher CPU and memory usage of the cloud server. Therefore, the increase in server power cancelled out the saved workload execution time. In summary, deploying MImageProcessor with strategy “2Medium” was more energy efficient than “1XLarge”.
2. Experimental results of RImageProcessor

We applied the same workload model and deployment strategies as MImageProcessor to compare the results of energy consumption and system throughput. In this set of tests, the size of the image file to process was set to 1000K. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was first deployed on one XLarge VM. This deployment strategy is named “1XLarge”. Then, we deployed the cloud application on two Medium VMs. This deployment is named “2Medium”. The application workload was evenly distributed across two Medium VMs. The energy consumption and system throughput of both MImageProcessor and RImageProcessor are presented in Figure 8.33 and Figure 8.34, respectively. We observed similar energy consumption and system throughput patterns as MImageProcessor.
Test set 8.5.2.3: In this test set, we keep the high-level workload model, while changing deployment strategy to allow resource over-commitment.

As we presented in Section 5.4.1.2, task memory usage has only slight impacted on total power consumption. In addition, we found that more memory allocated to ImageProcessor increased the system throughput of the cloud application, as described in test set 8.5.2.1. Thus, allocating more memory to ImageProcessor application helps improve its energy efficiency. However, memory resource on a cloud server is finite. Therefore, it is interesting to investigate the energy efficiency of the cloud application when server memory is over-committed. The experimental results of MImageProcessor and RImageProcessor are presented as follows:

1. **Experimental results of MImageProcessor**

In this set of tests, the size of image file to process was set to 2000K. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was firstly on two Medium VMs. This deployment strategy is...
named “2Medium”. As shown in Table 5.2, 4GB RAM was allocated to each Medium VM. The total amount of available physical memory on the cloud server was 10GB. Then, we increased the memory allocated to each VM from 4GB to 5GB. In this case, the physical memory had been fully occupied by the two VMs. This deployment strategy is named “2Medium_5GBRAM_Each”. At last, we increased the memory allocated to each VM from 5GB to 6GB. The total amount of memory allocated to the two VMs equalled to 12GB which had exceeded the amount of available physical memory of the cloud server. This deployment strategy is named as “2Medium_6GBRAM_Each”. The energy consumption and system throughput are presented in Figure 8.35 and Figure 8.36, respectively.

When we changed the deployment strategy from “2Medium” to “2Medium_5GBRAM_Each”, the energy consumption under both deployment strategies remained at the same level, as shown in Figure 8.35. The system throughput of “2Medium_5GBRAM_Each” was slightly higher than “2Medium”, as shown in Figure 8.36. The average difference in system throughput was less than 1% between the two strategies. This cloud application is a memory intensive one, whose system throughput is largely dependent on the size of its RAM. However, the available physical memory on the cloud server was 10GB, as presented in Table 5.2. The size of RAM allocated to the two Medium VMs was 8GB, which almost exhausted the RAM available on the physical server. Therefore, the system throughput improvement was not significant.

When we changed the deployment strategy from “2Medium_5GB_Each” to “2Medium_6GB_Each”, the energy consumption increased and system performance decreased, as presented in Figure 8.35 and Figure 8.36. When we allocated 6GB RAM to each VM, the memory resources were overcommitted as the total amount of virtual memory allocated exceeded the total amount of physical memory. Overcommitted resource allocation introduced synchronisation overhead on the physical server. Accordingly, the execution time of system increased, system throughput decreased and the energy consumption increased. This experiment result further validates the memory usage patterns presented in Table 6.3: for all live VMs running memory-intensive tasks, the total amount of virtual memory should be
equivalent to the total amount of available physical memory on the corresponding physical server.

![Figure 8.35 Energy Consumption of MImageProcessor with Different Memory Allocation](image)

Figure 8.35 Energy Consumption of MImageProcessor with Different Memory Allocation

![Figure 8.36 Throughput of MImageProcessor with Different Memory Allocation](image)

Figure 8.36 Throughput of MImageProcessor with Different Memory Allocation

2. Experimental results of RImageProcessor

In order to compare the results of energy consumption and system throughput to MImageProcessor, we applied the same workload model and deployment strategies as MImageProcessor. In this set of tests, the size of image file to process was set to 2000K. We increased the number of user requests per second from 10 to 40 in steps of 10. The cloud application was firstly on two Medium VMs. This deployment strategy is named “2Medium”. Then, we increased the memory allocated to each VM from 4GB to 5GB so that the physical memory of the cloud server had all been allocated to the VMs. This deployment strategy is named “2Medium_5GBRAM_Each”. At last, we increased the memory allocated to each VM from 5GB to 6GB, which was memory over-commitment scenario. This deployment strategy is named as “2Medium_6GBRAM_Each”. The energy consumption and system throughput of both MImageProcessor and
RImageProcessor are presented in Figure 8.37 and Figure 8.38, respectively. Similar energy consumption and system performance patterns as MImageProcessor can be observed.

8.6 Threats to Validity

Here we discuss the threats to the validity of the experimental results of the three case studies:

*Threats to construct validity.* The first main threat to the construct validity of the experimental results is the extraction of cloud applications’ workload models. A cloud application workload model is composed of a set of tasks. There are two aspects. The first one is the high-level workload model which defines the composition and execution sequences of the tasks. The second one is the low-level workload model which specifies detailed parameters of each user request. Thus, the main threat to construct validity is that whether the high-level workload model and
the low-level workload model can properly reflect the user behaviours of the target cloud applications. To minimise this threat, we first employed the high-level workload model based on empirical analysis or measurements of user behaviours. For example, the high-level application workload model for JPetStore is a sub model synthesised from the real-world JPetStore. Then we compared the experimental results of the modelled cloud application and the real-world cloud application. We found the similar energy consumption and system throughput patterns of the modelled cloud application and the real-world cloud application. By doing so, we could demonstrate that the workload models did reflect the user behaviours of the target cloud application.

The second main threat to the construct validity of the experimental results is that a single person derived the workload models, implemented StressCloud, designed all test cases, and carried out all the experiments and evaluations. Different person may have different point of view when carrying out the above mentioned activities. This may result in different energy consumption and throughput of cloud applications. To minimise this threat, the entire research approach presented in this thesis was reviewed and commented by our research team. In addition, most part of the research outcomes presented in this thesis have been published on International conferences. These publications have been reviewed and commented by peer reviewers as well. We are now applying StressCloud in other research projects. Thus, the construction validity of StressCloud can be further verified.

**Threats to external validity.** The main threat to the external validity of our experimental results is the representativeness of the selected cloud environment. We conducted the experiments on a HP Z400 server in SwinCloud. The diversity of real-world cloud servers can impact the power consumption of the server and the load test execution time. Thus, the energy consumption and system performance of the cloud application could be different from our experimental results in terms of actual figures. However, as the resources consumed by both modelled cloud application and real-world cloud application remain the same if we keep the workload model and deployment configuration the same, the patterns of energy consumption and system performance of the modelled cloud application and
real-world cloud application would still be similar to our experimental results in general.

*Threads to internal validity.* The main threat to the internal validity of our experimental results is the comprehensiveness of the experiments. We selected three representative cloud applications from different categories. For each selected cloud application in the tests, we applied a variety of deployment strategies and workload models, including high-level workload models and low-level workload models. The type of cloud application in real-world cloud environments can be very diverse. The energy consumption patterns of one cloud application can significantly differ from another cloud application, depending on the type and amount of resources required by the cloud application. It is impossible to evaluate the energy efficiency of all kinds of cloud applications. However, for each selected cloud application, no matter which workload model or deployment strategy we applied in the test, the energy consumption patterns of the modelled application were the same as the energy consumption patterns of the real-world cloud application. We believe that these experimental results are sufficient to demonstrate the practical value of our StressCloud approach.

### 8.7 Discussion

Throughout this chapter, we have analysed the trade-off between the energy consumption and system performance of three different types of cloud applications. A variety of application workloads and deployment strategies have been applied in the case studies. The experimental results show that the energy consumption and system throughput of cloud applications are highly related with application workloads and deployment strategies. Two energy consumption patterns presented in Chapter 6 have been validated in the case studies. These two energy consumption patterns are presented in Table 6.3 and Table 6.4, respectively. As we described in Section 7.3.4, StressCloud has the ability to validate the energy efficiency of a given workload model and a deployment strategy against the energy consumption patterns presented in Chapter 6. Before each test set, we ran the energy consumption pattern validation feature of StressCloud to pre-analyse the energy efficiency of the
workload model and deployment strategy which would be applied in the test. The validation results showed the same energy consumption and system consumption patterns analysed in test sets 8.3.2.1, 8.3.2.3, 8.4.2.2, and 8.5.2.3.

For both modelled and real-world cloud applications, we have used StressCloud to 1) model application workloads; 2) generate user requests; 3) run the load tests; and 4) collect performance and energy consumption data. For each modelled cloud application, we used StressCloud to deploy the service units required by the workload model to the target cloud environment. However, we deployed the real-world cloud application to the target cloud environment manually as the required service components of different real-world cloud application are different. We compared the experimental results of modelled cloud applications and real-world cloud applications. In general, the value of energy consumption of the modelled cloud application was higher than the corresponding real-world cloud application. In addition, the value of system throughput of the modelled cloud application was lower than the corresponding real-world cloud application. The differences between the modelled cloud application and the real-world cloud application imply that the models of the cloud application need to be improved to be more realistic. As we analysed in test set 8.3.2.1, the modelled cloud application does not support caching mechanism, which results in longer average response time of user requests. In order to address this issue, we are extending the workload models and StressCloud to support caching mechanisms. However, the patterns of energy consumption and system throughput of both the modelled and the corresponding real-world cloud application were the similar, which demonstrates the practical value of our StressCloud approach.

We have validated that cloud server resource over-commitment can negatively affect the energy efficiency of cloud applications. Due to the fact that the physical resources are usually finite, it will be interesting to develop a model which indicates different levels of cloud application energy efficiency. This energy efficiency model can be further applied to design algorithms to develop equilibrium between available resources and energy efficiency. These algorithms can be applied to accurately predict the resources needed per application load which can directly improve the energy efficiency of cloud applications.
The storage devices installed on the server we adopted in our case studies were hard disk drives (HDD), as presented in Table 5.2. We investigated the energy efficiency of different file I/O operations on an HDD in Section 8.4. HDDs have been the predominant storage devices for cloud servers for a long time. Although HDD technology has improved dramatically in the years, there are still internal limitations to the response times (access time and latency) that an HDD can achieve. These limitations and the continuing demand for better response times and more throughput have created the need for a different storage technology that avoids seeks and rotational delays. This technology, known as Solid State Drives (SSD), can provide access times measured in microseconds. In addition, SSDs consume less power than HDDs [132]. However, with the same amount of storage capacity, SSDs are more expensive than HDDs. Therefore, it will be interesting to evaluate the cost efficiency of HDDs and SSDs with data-intensive workloads. These investigation results can be used as guidelines for cloud providers to either extend existing data centres or build new data centres.

8.8 Summary

In this chapter, we presented three case studies of using StressCloud to analyse system performance and energy consumption. We introduced a set of case studies to 1) further validate StressCloud; 2) further validate the energy consumption signatures; and 3) show how to use StressCloud to further analyse the trade-off between system performance and energy consumption of different cloud applications.

We first gave an introduction of all three case studies. Then we presented the experimental setup adopted in the case studies. Then, for each case study, we 1) presented the test case design; and 2) analysed the experimental results. Finally, we drew conclusions from the experimental results in the three case studies. Our experimental results showed that the system performance and energy consumption of cloud applications are closely related to cloud system configurations and application workloads. The experimental results also validated the practical value of our StressCloud approach.
Chapter 9

Conclusions and Future Work

This chapter highlights the main findings of the research work on automating energy efficiency analysis for cloud applications presented in this thesis. It also discusses future research directions.

9.1 Key Contributions of the Research

The significance of this research is that we have introduced a set of solutions for accelerating the energy efficiency evaluation process of cloud applications with different deployment strategies. The major contributions of this thesis are as follows:

- A novel energy consumption model has been designed for profiling and analysing the energy consumption of cloud applications. This new energy consumption model provides a basis for characterising the energy consumption of cloud applications under different system configurations and application workloads. Most existing energy consumption models take individual hardware components as the fundamental units for energy profiling. Our energy consumption model considers a single task running on the cloud system as the fundamental unit for energy profiling. It takes into account the types of cloud applications and their workloads, as well as the system configurations.

- Extensive experiments have been conducted to profile and investigate system performance and energy consumption of different types of cloud applications with varying workloads and system configurations. The energy efficiency of single cloud application, as well as multiple cloud applications simultaneously running on the same server/VM has been investigated. The
experimental results demonstrated the correlation coefficients of energy consumption, system configurations and workloads, as well as the performance of the cloud applications. In addition, a set of guidelines have been derived from the experimental results. These guidelines can be adopted to achieve improved energy efficient deployment, resource provisioning and management strategies for cloud applications.

- A series of novel and extensible energy consumption patterns from our experimental results have been designed and formalised as “signatures” using the Object Constraint Language (OCL). In order to formalise these energy consumption patterns, the following schema and models are developed: 1) a system performance and energy consumption pattern definition schema that captures the details of a given system performance and energy consumption evaluation scenario, including cloud application workload, cloud system configurations, required resources and application deployment strategies; 2) a cloud system architecture model that describes the architectures and system configurations of cloud systems; and 3) a workload model that captures the details of cloud applications’ workloads. System architects and performance engineers can employ these energy consumption patterns to predict the energy efficiency of a target cloud application without running load tests. These patterns can be applied to different cloud applications in different cloud environments as they are generic and independent on cloud platforms.

- A novel and automatic performance and energy consumption analysis tool for cloud applications in real-world cloud environments named StressCloud have been designed and implemented. The key contributions of StressCloud are: 1) supporting user-defined high-level architecture and workload models for complex cloud applications; 2) supporting energy efficiency validation of complex cloud applications using formalised “signatures” based on high-level architecture and workload models; 3) fully automatic generation and deployment of large-scale cloud application workload test services and cloud application model prototype implementations; 4) the ability to realistically stress test existing cloud applications and potential cloud
application models; 5) automatic system performance and energy profiling of cloud applications; and 6) analytical support for pre-test model energy and performance weaknesses and post-test energy and performance metric analysis.

- Extensive validation of StressCloud on exemplar cloud applications, models and workloads have been conducted. The energy efficiency of different types of cloud applications for both modeled cloud applications and corresponding real-world applications have been analysed. With the same workload models and cloud architecture models, consistent energy consumption and system performance variations of both modeled and real-world cloud applications have been observed in all case studies, which demonstrates the practical value of StressCloud.

### 9.2 Future Work

In the future, further investigation into energy efficiency of cloud applications can be carried out in the directions described below.

The cross-server scheduling and communication overhead in one cloud environment can be significantly different from another, depending on the scheduling mechanism adopted by the cloud service providers and the types and distribution of the constituent servers. Thus, the energy consumption patterns of cloud applications deployed on multiple physical servers can be significantly different from on a single physical server. We plan to conduct extensive experiments with heterogeneous models of cloud servers to further analyse the energy efficiency of cloud applications deployed across multiple cloud servers. This will help performance engineers and system architects understand the impacts of different scheduling mechanisms on system performance and energy consumption of cloud applications.

According to [98], a server with zero workload consumes about 60% of its peak power, which contributes to a large proportion of energy consumption in data centres as discussed in Chapter 4. A representative approach on energy saving in the cloud environments is load balancing between servers and suspension of idle
servers. With idle servers switched off, one can optimise resource usage and reduce energy consumption by over 70% [133]. We plan to look into the optimisation of cloud application placement and dynamic load balancing across multiple servers. The objective is to minimise the number of physical servers allocated to the cloud application while still guaranteeing that SLAs are met.

We plan to extend StressCloud to support more cloud platforms. Currently, it only supports VMWare as the hypervisor. We will create a set of templates to traverse the workload and cloud architecture model in order to generate load test scripts and deployment scripts for the selected cloud platform. This will allow a system architect or performance engineer to add more templates to StressCloud to support more cloud platforms. We also plan to extend StressCloud to support reverse-engineering of user behaviour models and architecture models. This will allow a system architect or performance engineer to reverse-engineer an existing web application behaviour model and architecture model, and then map the workload model and the architecture model to different types of tasks in a real-world cloud environment. In this way, the system architect or performance engineer will be able to analyse the performance and energy consumption of the cloud application after its migration to another cloud environment.

StressCloud is realised as a set of Eclipse IDE plug-ins. All model components are implemented as graphic elements using Eclipse Graphic Modelling Framework (GMF). In order to use StressCloud, the system architect or performance engineer needs to install desktop Eclipse IDE and related plug-ins on local machine. We plan to restructure StressCloud a cloud-based IDE so that users can access it through a web browser. This will free the system architect or performance engineer from tedious installation process and hardware performance concerns. Eclipse Orion\(^\text{18}\) can be adopted to re-build StressCloud. It is a cloud-based IDE and allows third-party developers to extend the behaviour of the Eclipse Orion editor on the fly by installing plug-ins.

We are going to extend both the high-level and low-level cloud application workload models to support caching mechanism so that the modelled cloud application is more realistic. This also allows the system architect and performance

\(^{18}\) www.eclipse.org/orion
engineer to compare the energy efficiency of different application architectures, i.e. with or without caching mechanism. We also plan to extend the cloud application workload models to support greater range of more complex cloud applications, e.g., cloud applications that perform more sophisticated computation-intensive and data-intensive tasks.
Reference


[71] Poess, M., and Nambiar, R.O.: ‘Tuning servers, storage and database for energy efficient data warehouses’. in. the 26th International Conference on Data Engineering (ICDE2010), Long Beach, CA, USA, 2010, pp. 1006-1017.


