



Libraries and Learning Services

University of Auckland Research Repository, ResearchSpace

Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognize the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the [Library Thesis Consent Form](#) and [Deposit Licence](#).

UNIVERSITY OF AUCKLAND

Optimal Orchestration of a Cloud-Based Internet Cafe

by

Isaac Hamling

A thesis submitted in fulfillment of the requirements for the
degree of Doctor of Philosophy in Operations Research

in the

Faculty of Engineering
Department of Engineering Science

February 2019

UNIVERSITY OF AUCKLAND

Abstract

Faculty of Engineering
Department of Engineering Science

Doctor of Philosophy

by [Isaac Hamling](#)

supervised by Michael O'Sullivan and Cameron Walker

The idea to use cloud computing and virtualisation to supply video games to end users known as cloud gaming has been growing in popularity. Internet cafes are one potential application for cloud gaming. Internet cafes are a large market in China which has over 185,000 internet cafes each with an average of 120 seats. These internet cafes have significant resource inefficiency which can be improved using a cloud gaming model to supply internet cafe users. A system is presented for a cloud-based internet cafe and algorithms for maximising resource utilisation in cloud gaming.

The cloud-based internet cafe model replaces the traditional desktop computers in an internet cafe with servers, thin clients, and virtual machines with specifications designed to meet specific user demands. Virtual machines are run on the servers to supply internet cafe users with their desired game or service. Important decisions need to be made: which users to accept, and upon which servers they should be placed. This problem is the cloud gaming resource allocation problem.

An integer programming model is formulated for solving the offline cloud gaming resource allocation problem for cloud-based internet cafes. This offline model shows that moving to a cloud-based internet cafe improves daily profits over the current zoned internet cafe model utilised in China and significantly improves resource utilisation.

A further three algorithms are presented to solve real world demand in a cloud-based internet cafe. A prebooking system for solving a semi-online version of the problem. This algorithm is used to place users who book seats in the cloud-based internet cafe in advance. Online greedy and competitive algorithms are also presented for solving the online resource allocation problem for cloud-based internet cafes. All three algorithms show competitive performance when compared to the offline optimal allocations with exact performance depending the user demand profile.

The cloud-based internet cafe shows the ability to improve profits, and resource efficiency in internet cafes. The four algorithms developed can be applied to different types of internet cafes. The competitive algorithm performs best in busy internet cafes, while the greedy algorithms are better when they are less busy. The prebooking system is most useful for an internet cafe with regular customers. While the offline integer program can be used for future planning or planned events. All four algorithms and cloud gaming hardware can combine to form a cloud-based internet cafe system for building and operating a cloud-based internet cafe.

Acknowledgements

First of all, thank you to my supervisors Dr Micheal O’Sullivan and Dr Cameron Walker for all the help and support throughout my doctorate. Extra thanks to Mike and Cam for the job opportunity when funding ran out so I could continue to work on my thesis.

Thanks to the ORUA research group for all the help. Thanks to Ola for helping out over summer testing remote desktop connections. Special thanks to Tim Harton for being my PhD buddy and all the help setting up the cloud hardware, software and networking for testing all our different cloud projects. Extra thanks to Tim and his wife Dr Kat Gilbert for being good friends. Thank you to Citrix for providing software licenses for testing the hardware implementation.

Thanks to our colleagues in China for the opportunity to visit and see the business implementation of virtualisation in internet cafes. Special thanks to Felix Xia and William for organising accommodation, transport and showing me around their workplace.

Thanks to the OptAli research group for the opportunity to travel overseas and collaborate with other universities. Thanks to Dr Andrea Raith and Dr Andrew Mason for organising the OptAli project at the University of Auckland and giving me the opportunity. To the staff and students I met at the universities in Europe, thank you for being friendly and welcoming. Special thanks to Professor Jesper Larsen at Denmark Technical University for helping me find my way in Denmark and for the research support. Thanks to Professor Sven O Krumke and Junior Professor Clemens Thielen at the Technical University of Kaiserslautern for helping me in Germany and for the research ideas. Additional thanks to Clemens for the research support in online algorithms and for the shared publications. Thanks to Oscar Dowson for keeping me company while travelling in Europe.

Thanks to my family for supporting me throughout my doctorate. My direct family mum, dad and brother, thanks for all the love, support, and good conversation. Auntie Anne and Uncle Ian, Auntie Dianna and Uncle Wayne, Hannah and Nick, Ben and Sora, Tracy, Jack and children, thanks to all of you for the meals, company, and for being kind, and caring while I have lived in Auckland. Thanks to my remaining family, Uncle John and Kristen, Nana and Granddad, Grandma and Granddad Bob, Tania and Miles for the long-distance support and company on Christmas and holidays.

Thanks to all my friends for keeping me company and helping out with everything. All the people who’ve lived with me, Brent, Tom, Fraser, Rothborey, Llew, Andrew, and Rachel thanks for all the love and support and not being too messy. Special thanks to my close friends Darren, Fraser, Rothborey, Llew, and Leah for spending time with me and helping to

take my mind of everything going on. Thanks to my friends overseas, Oli, Blanca, and Josh for letting me stay with them while I was travelling, keeping me company, and showing me what their lives are like.

Finally thanks to Meg, and her family for letting me stay with them and being welcoming to someone they had never met before. Special love to Meg for being awesome but also for believing in me and being supportive of my abilities.

Contents

Abstract	iii
Acknowledgements	v
List of Figures	xi
List of Tables	xiii
Abbreviations	xv
Notation	xix
Co-Authorship Forms	xxiii
1 Introduction	1
1.1 Motivation	1
1.1.1 Internet Cafes	2
1.1.2 Cloud Gaming	5
1.2 Cloud-Based Internet Cafes	7
1.2.1 Virtualisation	8
1.2.2 Virtual graphics processing units (GPUs)	8
1.2.3 Combining Cloud Gaming and Internet Cafes	9
1.2.4 Optimisation	10
1.2.5 Using Optimisation in Cloud-based Internet Cafes	12
1.3 Thesis Outline	14
1.4 Contribution	15
2 Background	17
2.1 Internet Cafes	18
2.2 Virtualisation	20
2.2.1 Building a Cloud	21
2.2.2 Supplying Cloud Services	23
2.2.3 Virtual Desktop Infrastructure	24
2.2.4 Cloud Gaming	24
2.2.5 Virtual GPUs (vGPUs)	26

2.3	Resource Allocation in Cloud Computing	27
2.3.1	Integer Programming	30
2.3.2	Competitive Analysis of Online Algorithms	33
2.3.3	Online Algorithms and Heuristics	34
2.4	Literature Review	35
2.4.1	Cloud virtual machine (VM) Placement	36
2.4.2	Cloud Gaming VM Placement	42
3	Building a Cloud-Based Internet Cafe	47
3.1	Hardware	48
3.1.1	Servers	50
3.1.2	GPU	50
3.1.2.1	virtual GPUs (vGPUs)	52
3.1.3	Storage	56
3.2	Hypervisors	58
3.2.1	Citrix XenServer	58
3.2.2	Other Hypervisors	59
3.3	Remote Desktop	59
3.3.1	Citrix XenDesktop	60
3.3.2	RemoteFX	62
3.3.3	Steam Streaming	62
3.3.4	Other Remote Desktop Software	63
3.4	Environment Testing	64
3.4.1	Single Machine Testing	65
3.4.2	Parallel Testing	68
3.4.3	Summary	70
4	Allocation Problem	73
4.1	Problem Description	73
4.2	Models	77
4.2.1	Notation conventions	77
4.2.2	Inputs	77
4.2.2.1	Services	78
4.2.2.2	Servers	78
4.2.2.3	VMs	79
4.2.3	Configurations	80
4.2.3.1	Users	81
4.2.4	Decisions	82
4.2.5	Constraints	82
4.2.6	Outputs	83
4.3	Test Data Set	83
4.3.1	First Test Set	84
4.3.2	Services offered	84
4.3.3	Server Inputs	85
4.3.4	User Inputs	86
4.3.5	Second Test Set	89

4.3.6	Services offered	90
4.3.6.1	Server Inputs	91
4.3.6.2	User Inputs	92
4.3.7	Generation Process	96
4.3.8	Test Profiles Summary	97
5	Offline Integer Program	99
5.1	Initial Model	100
5.1.1	Decision Variables	100
5.1.2	Integer Program	102
5.1.2.1	User constraints	102
5.1.2.2	Server constraints	103
5.1.2.3	Event constraints	104
5.2	Efficiency Improvements	105
5.2.1	Symmetry	105
5.2.2	Memory Use	107
5.3	Updated Model	108
5.3.1	Decision Variables	108
5.3.2	Integer Program	109
5.3.2.1	User Constraints	110
5.3.2.2	Supply Constraints	110
5.4	Fixed Zone Model	111
5.4.1	Decision Variable	112
5.4.2	Integer Program	112
5.5	Results	113
5.5.1	Test Set One	113
5.5.2	Test Set Two	118
6	Prebooking Integer Program	121
6.1	Model	123
6.1.1	Batch Solver	123
6.1.2	Model Definition	125
6.1.2.1	Decision Variables	125
6.1.2.2	Integer Program	126
6.2	Results	127
6.2.1	Time Horizon	128
6.2.2	Solve Time	131
6.2.3	Summary	132
7	Online Algorithm	135
7.1	Greedy Algorithms	136
7.2	Competitive Algorithm	137
7.2.1	Algorithm	139
7.2.2	Proofs	141
7.2.3	Grouped Users	149
7.2.4	Aggressive Improvement	150
7.3	Results	151

7.3.1	Internet Cafe Size	152
7.3.2	Stay Duration	154
7.3.3	Comparison	156
8	Discussion and Conclusion	159
8.1	Comparison	160
8.2	Internet Cafe Application	161
8.3	Graphics Driven Cloud Application	162
8.4	Business Application	163
8.5	Education Application	164
8.6	Conclusion	165
8.6.1	Key Outcomes	165
8.6.2	Future Work	167
8.6.3	Application	168
8.6.4	Research Questions	169
A	Configurations	171
	Bibliography	173

List of Figures

1.1	Example set up and seating arrangement of a zoned internet cafe	4
1.2	Example cloud-based internet cafe showing seating flexibility	13
3.1	Test Cloud-Based Internet Cafe Setup	49
3.2	Nvidia GRID K1 card layout and specifications	51
3.3	Nvidia GRID K2 card layout and specifications	52
3.4	Types of vGPU with random access memory (RAM) and core divisions . . .	53
3.5	Invalid VM additions to GRID K2 card	54
3.6	Valid VM additions to a GRID K2 card. Note that the K240 VM on the K260 card will have higher than specified graphics performance	55
3.7	Grid of 3DMark scores for vGPU and central processing unit (CPU) configurations. Tests from left to right bars: Ice Storm; Cloud Gate; Sky Diver; and Fire Strike.	67
4.1	Probability density function for hours users arrive in test set one	87
4.2	Probability density function for duration of stay in test set one	88
4.3	Probability density function for hours users arrive in test set two	93
4.4	Probability density function for duration of stay in test set two for 2.5 hour average	94
4.5	Probability density function for duration of stay in test set two for 3.5 hour average	95
4.6	Probability density function for duration of stay in test set two for 4.5 hour average	95
4.7	Probability density function for duration of stay in test set two for 6.5 hour average	96
5.1	Utilisation of CPU, RAM, and GPU resources over time for test set one test 2	116
5.2	Utilisation of CPU, RAM, and GPU over time for test set one test 6	117
5.3	Comparison of average utilisation of resources using cloud-based vs. traditional internet cafe	118
6.1	Features of a prebooking system over time from when a user places a booking	122
6.2	Average percentage difference of batch solver from optimal for the three different numbers of users	130
6.3	Average percentage difference of batch solver from optimal for the four different stay duration's	131
6.4	Average percentage difference of batch solver from optimal with 10 second vs 120 second solve time	133

7.1 Percentage From Optimal for Online Algorithms (average for all realistic and stress test cases)	156
---	-----

List of Tables

2.1	Summary of key VM placement papers	39
2.2	Summary of cloud gaming VM placement papers	43
3.1	Specifications of servers used for feasibility testing	50
3.2	vGPU names and specifications	52
3.3	GPU game performance with various settings	57
3.4	Summary of remote desktop software performance	60
3.5	Test results for the K140 vGPU (Percentages show difference from single machine)	69
3.6	Test results for the K240 vGPU (Percentages show difference from single machine)	69
3.7	Test results for the K260 vGPU (Percentages show difference from single machine)	69
3.8	3DMark scores of K1 and K2 vs Gaming PC	71
4.1	Example VMs for a K2 server	75
4.2	Example of configurations for a K2 server	75
4.3	Profit and minimum 3DMark scores for test set one services	85
4.4	Test set one Internet Cafe Sizes and Server Quantities	85
4.5	Test set one virtual machine specifications	86
4.6	Proportion of demand for each service for test set one	89
4.7	Profit and performance details for test set two services	90
4.8	Test set two Internet Cafe Sizes and Server Quantities	91
4.9	Test set two virtual machine specifications	92
4.10	Proportion of demand for each service for test set two	93
4.6	Proportion of demand for each service for test set one	114
5.1	Comparison of results for test set one with 100 users after running each test for ten minutes	114
5.2	Comparison of results for test set one with 500 users after running each test for ten minutes	115
5.3	Comparison of results for test set one with 1000 users after running each test for ten minutes	116
5.4	Comparison of results for test data set two with 10 minute solve times versus 10 hour solve times	119

6.1	Difference between optimal objective and batch solver results averaged over all tested time horizons and for the 2 hour time horizon over all test sets (10 problem instances for each number of users)	129
6.2	Comparison of 10 seconds and 120 second solve times for average objective from batch solver	132
7.1	Competitive ratio and calculation components for different user stay durations. Symbols, λ , F , $ Q $, J , and μ are defined in §7.2.1	152
7.2	Greedy algorithm performance for different size internet cafes, average stay = 2.5 hours with 95% confidence intervals	154
7.3	Competitive algorithm performance for different size internet cafes, average stay = 2.5 hours with 95% confidence intervals	154
7.4	Greedy algorithm performance for different average stay durations, internet cafe size = 500 with 95% confidence intervals	155
7.5	Competitive algorithm performance for different average stay durations, internet cafe size = 500 with 95% confidence intervals	155
7.6	Results and Comparison for 20 Seat Internet Cafe	157
8.1	Comparison of algorithm performance	160
A.1	K1 server configurations for test set 1	171
A.2	K2 server configurations for test set 2	172

Abbreviations

API application programming interface

CAD Computer Aided Design

CDN content delivery network

CI Confidence Interval

CPU central processing unit

DDR double data rate

DHCP dynamic host configuration protocol

DNS dynamic name server

DVI display visual interface

FIFO First-In First-Out

FPS frame per second

GB gigabyte

GDDR graphics double data rate ([DDR](#))

GPU graphics processing unit

GUI graphical user interface

GaaS Gaming as a Service

HDD hard disk drive

HDMI high definition multimedia interface

ICT information and communications technology

IP integer program

IaaS Infrastructure as a Service

LAN local area network

NAS network-attached storage

NFS Network File System

OS operating system

PC personal computer

PaaS Platform as a Service

RAM random access memory

RDP remote desktop protocol

SAN storage area network

SFP+ small form-factor pluggable transceiver plus

SLA service level agreement

SSD solid state drive

SSH secure shell

SaaS Software as Service

UI user interface

VDI virtual desktop infrastructure

VGA video graphics array

VM virtual machine

VNC Virtual Network Computing

VPN virtual private network

VRAM video random access memory

WAN wide area network

vCPU virtual [CPU](#)

vDisk virtual disk

vGPU virtual [GPU](#)

vRAM virtual [RAM](#)

vSwitch virtual switch

Notation

S	List of servers
s	A server
Q	List of server resources
q	A server resource
M	List of GPU core configurations
m	A GPU core configuration
U	List of internet cafe users
u	An internet cafe user
R	List of services offered
r	A service offered
$r(u)$	The service a user demands
G	List of GPU core types
g	A GPU core type
$g(s)$	The type of GPU core a server has
$g(s, r)$	The type of GPU core a VM would require on a server for a service
$g(m)$	The type of GPU core a configuration runs on
T	List of time points
$T(u)$	List of time points when a user is active
t	A point in time

E	List of event points when users arrive
e	A time point when a user arrives
e^+	The event point after e
e^-	The event point before e
$Z(g)$	List of vGPU core types for a specific GPU core type
ζ	A vGPU core type
$\zeta(s, r)$	The type of vGPU core a VM would require on a server for a service
$\zeta(m)$	The type of vGPU core a configuration uses
$z(r)$	A list of servers which can supply a service
$z(u)$	A list of servers a user can be placed onto
$c(s, q)$	Quantity of a resource available on a server
$b(s, r, q)$	The quantity of a resource consumed by a VM on a server for a resource and service
$b(s, q, u, t)$	The quantity of a resource consumed on a server by a user at a time
$\rho(m, q)$	The quantity of a resource consumed by a configuration
$K(s)$	List of indexes for GPU cores on a server
k	A GPU core index
$\pi(r)$	Profit per time period of a service
$p(u)$	The profit available from a user
$\bar{n}(s, r)$	The number of users a VM can support on a server for a service
\bar{N}	The number of seats in an internet cafe
$v(m, r)$	Number of VMs running for a service for a configuration
$a(u)$	The time a user arrives
$d(u)$	The time a user departs
$j(u)$	The duration a user stays

J	The maximum duration any user stays
$\nu(s, e, r)$	The number of VMs running for a service on a server at an event
δ	The frequency with which the problem is solved
σ	The quantity of future information given
γ	The amount of time before arriving users are informed of acceptance
$i(s, q, t)$	Quantity of a resource being consumed on a server at a time
$L_{s,q,u,t}$	The normalised load on a server for a resource at a time when a user arrives
$V_{s,q,u,t}$	The exponential load on a server for a resource at a time when a user arrives
μ	A constant for the competitive algorithm
λ	A constant for the competitive algorithm
κ	A constant for the competitive algorithm
F	A constant for the competitive algorithm
A	A list of users accepted by both competitive algorithm
H	A list of users rejected by the competitive algorithm but accepted by the optimal solution

Co-Authorship Forms

Co-Authorship Form

Graduate Centre
The ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Competitive Online Algorithms for Allocating Cloud-Based Internet Cafe Resources

Chapter 7

Nature of contribution by PhD candidate	Methodology, Implementation, Test data, Results, Writing
Extent of contribution by PhD candidate (%)	>80%




CO-AUTHORS

Name	Nature of Contribution
Clemens Thielen	Model development, discussion, comments, reviews, and editing
Michael O'Sullivan	Supervision, discussion, comments, reviews, and editing
Cameron Walker	Supervision, discussion, comments, reviews, and editing

Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Clemens Thielen		04.12.2017
Michael O'Sullivan		14/12/17
Cameron Walker		14/12/17

Co-Authorship Form

Graduate Centre
 The Clock Tower – East Wing
 22 Princes Street, Auckland
 Phone: +64 9 373 7599 ext 81321
 Fax: +64 9 373 7610
 Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Improving Resource Efficiency in Internet Cafes by Virtualization and Optimal User Allocation

2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)

Chapter 4: Section: 2, and 3.1. Chapter 5: Section: 1, and 5.1

Nature of contribution by PhD candidate	Methodology, Implementation, Test data, Results, Writing
Extent of contribution by PhD candidate (%)	>80%

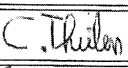
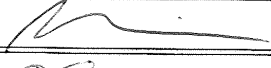

CO-AUTHORS

Name	Nature of Contribution
Clemens Thielen	Discussion, comments, reviews, and editing
Michael O'Sullivan	Supervision, discussion, comments, reviews, and editing
Cameron Walker	Supervision, discussion, comments, reviews, and editing

Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Clemens Thielen		04.12.2017
Michael O'Sullivan		14/12/17
Cameron Walker		14/12/17

Dedicated to Mum

Chapter 1

Introduction

This thesis presents the concept of cloud-based internet cafes, their architecture, and methods to ensure such cafes are effectively utilised. The concept of cloud-based internet cafes is motivated by inefficiencies in the typical architecture of existing internet cafes and the development of new technology which enables cloud gaming. This thesis presents a prototype cloud-based internet cafe which leverages virtualisation technology. The information about this prototype is then combined with internet cafe user surveys to build a representative data set for a cloud-based internet cafe's users. Once the cloud-based internet cafe and users have been characterised, resource allocation algorithms are developed which allocate resources to internet cafe users so that their desired services are provided efficiently in relevant scenarios. The algorithms presented are both offline and online algorithms.

1.1 Motivation

Internet cafes and cloud gaming are two similar information and communications technology (ICT) paradigms that supply games as a service to paying users. An internet cafe requires a person to access the service in person, and a cloud gaming service supplies games remotely over the internet. Internet cafes suffer from inefficiency because they use fixed personal

computers (PCs) to service a variety of user demands. Cloud gaming makes use of servers with significantly more flexibility, but requires intelligent algorithms to utilise server resources efficiently. New technology for GPU virtualisation offers increased flexibility in the resources supplied by servers and, with intelligent decision making, the ability to improve resource usage efficiency for both internet cafes and cloud gaming. However, cloud gaming suffers from latency issues as a result of the service being provided over the internet, i.e., wide area networks (WANs). Since internet cafes provide games directly on fixed PCs, latency is not an issue for gaming within an internet cafe.

This thesis addresses the following research questions, which are fundamental in the development of a cloud-based internet cafe:

1. can cloud gaming be combined with internet cafes, to create a new system known as a cloud-based internet cafe, which alleviates the inefficiencies of both internet cafes and cloud gaming?
2. can optimisation algorithms provide methods for the efficient allocation of cloud resources in a cloud-based internet cafe, particularly given the use of GPU resources?

To the best of the author's knowledge no one has presented the concept of a cloud-based internet cafe in previous research, created a representative data set of such an internet cafe's users, or developed offline and/or online algorithms for the corresponding cloud resource allocation problem that includes the use of virtual graphics processing units (vGPUs) for cloud gaming.

1.1.1 Internet Cafes

Internet cafes are a major industry world wide and can be found in any major city. In China there are more than 185,000 internet cafes averaging 120 seats per cafe [1]. An internet cafe provides computing resources as a service to users by: 1) providing a space containing computers (usually leased by the cafe); and 2) leasing time on these computers to users.

These computers offer services that include playing video games, web browsing, email access, and watching videos.

In first world countries internet cafes are used as social spaces to play video games with friends or to host tournaments [2, 3]. In China, South East Asia, and Korea there is a significant gaming culture surrounding internet cafes [4].

In third world countries where people have limited access to PCs and the internet, an internet cafe offers a place to access these resources. These internet cafes provide entertainment in the form of playing games or watching videos, they give access to news and email, and enable users to chat online with friends [5]. Nigeria holds the Guinness world record for the largest internet cafe in the world with 1,027 seats [6].

Most internet cafes' services include a number of different games. The computers in the internet cafe must be able to run all offered services. Often the hardware (resource) requirements in terms of CPU, RAM, and GPU vary greatly between the different services. For example, browsing the web and checking emails require very little CPU and GPU compared to playing games. Different games also have a large variance in resource requirements. The most popular games are low requirement, competitive, multiplayer games including League of Legends, DoTA2, Counter-Strike: Global Offensive, and Overwatch. However, an internet cafe will also offer the latest games including Fallout 4, Call of Duty WWII, and Battlefield 1 which have significantly higher resource requirements. The computers that the internet cafe leases must be able to run the latest games, but many users will demand web browsing, emails, or games with low resource requirements. When using any service other than the latest games the computer will have a quantity of CPU, RAM, and GPU which is left unutilised.

This gap between the resources required and the resources available results in an inefficiency. The internet cafe is paying for computing resources which are not being utilised. Chinese internet cafes address this inefficiency by dividing the internet cafe into two zones. One zone is specialised for gaming, and the other is specialised for web browsing, web games, email, and watching videos. This reduces the inefficiency in an internet cafe, but also reduces the

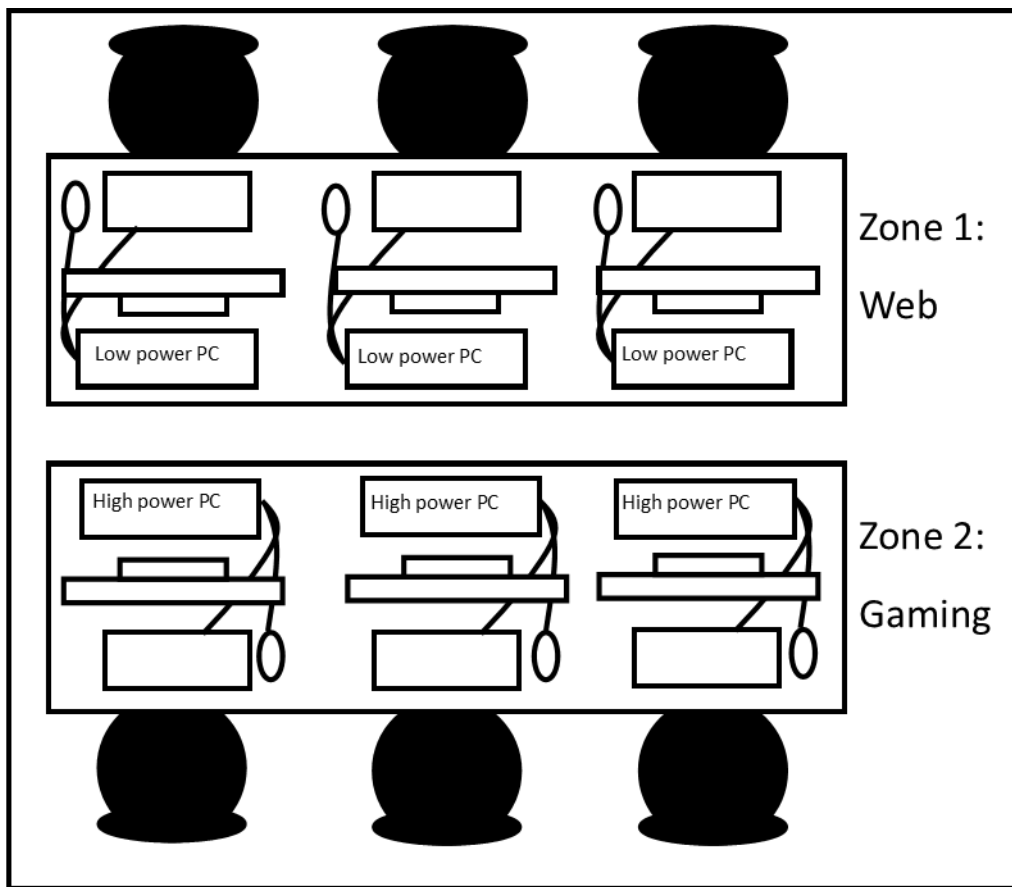


FIGURE 1.1: Example set up and seating arrangement of a zoned internet cafe

cafe's flexibility by limiting the quantity of each service offered. In these zoned internet cafes, each seat is restricted to supplying only a subset of the services offered. This restriction reduces profits when all seats of one service are filled and the internet cafe is forced to turn away new users requesting those services. Figure 1.1 shows a small example of how a six seat internet cafe may be divided into zones. One side has computers which meet the requirements of the web level services and the other side has more powerful computers which meet the requirements of the latest games. In this internet cafe once there are three gaming users the internet cafe is forced to turn away any future gaming users losing the potential profit from these users.

Even with this two zone model an internet cafe continues to have significant inefficiency due to the wide range of requirements for the games being offered. Fixing this inefficiency would require more zones, further reducing the flexibility of the internet cafe. For this reason an

internet cafe would greatly benefit from a system which could have an increased number of zones without fixing the number of seats allocated to any one zone. This system would be able to reduce inefficiency without significantly impairing flexibility. In order to implement a flexible, multi-zone system the internet cafe paradigm would need to shift away from physical computers placed at each seat and into a cloud computing system remotely supplying resources to each seat. Before describing this new internet cafe paradigm in §1.2 we describe the cloud technology which enables this paradigm, cloud gaming.

1.1.2 Cloud Gaming

Cloud gaming or Gaming as a Service ([GaaS](#)) is a service which offers computing resources for playing video games. A cloud gaming provider rents gaming resources to users who pay for the time they use the resources.

Cloud gaming technology supplies games to users over the internet in the form of either: a Windows operating system ([OS](#)) onto which users can install their own games; or direct connections to a specific game. Cloud gaming removes the need for a home [PC](#) with the ability to play the latest games. Current notable cloud gaming platforms include Sony Playstation Now[7], Nvidia GeForce Now[8], and LiquidSky[9].

Cloud gaming is a specific service within cloud computing. Cloud computing is a service which offers remote computing resources for a variety of tasks. Like cloud gaming, cloud computing has a cloud provider offering computing resources for rental by users. These users pay for the time and quantity of resources they use.

There are three ways the cloud is offered: Infrastructure as a Service ([IaaS](#)), Platform as a Service ([PaaS](#)), and Software as Service ([SaaS](#)). Under [IaaS](#) the cloud provider offers raw computing resources for use by users, who are free to install any [OS](#) and software of their choice. Examples of [IaaS](#) include Amazon Web Services EC2 [10], and Microsoft Azure [11]. Under [PaaS](#) the cloud provider offers preconfigured computing resources with [OS](#), drivers, and networking fully set up. An example of [PaaS](#) is Amazon Web Services Elastic Beanstalk

[10]. Under [SaaS](#) the cloud provider offers only a software application for use by the customer. Examples of [SaaS](#) include Google Apps, such as Google Docs [12], and Netflix [13].

Cloud gaming providers supply [GaaS](#) under either the [PaaS](#) model (supplying a Windows [OS](#) for playing games on) or [SaaS](#) (supplying specific games).

A cloud provider can allocate each user the exact resources they require for the service they demand. In order to ensure that the available resources are utilised efficiently, cloud providers use resource allocation algorithms. Resource allocation algorithms are designed to allocate a cloud provider's resources to users in such a way as to maximise the total profit from the users while meeting the users' demand for services. The cloud's flexibility enables the offering of continuous resource quantities for a wide variety of services.

One key hardware feature of cloud gaming is the inclusion of [GPUs](#) in a cloud's resources. The [GPUs](#) that are included can be either standard desktop [GPU](#) cards, or special virtualisable [GPUs](#). These [GPU](#) resources enable the games running in the cloud to have the enhanced graphics performance required to ensure satisfactory game play.

Extensive research has been conducted into approaches that efficiently run cloud computing environments, but there is little research into efficiently running cloud gaming environments. A cloud gaming environment is different from standard cloud environment due to: 1) the addition of [GPU](#) cards to the cloud resources; and 2) the more consistent demand for resources of gaming users because they require fixed resources to play a game (other users' demand for resources can vary depending on what they are using the demanded service for) .

The addition of [GPU](#) virtualisation to the cloud gaming space offers a significant increase in flexibility for allocating users to cloud resources. In order to make use of this increased flexibility in an efficient way, resource allocation algorithms are required for placing users. While these algorithms have been developed for cloud computing, research is needed for user placement in a cloud gaming environment. New algorithms are required to handle the complexities of [vGPUs](#) which have unique restrictions on how they are virtualised, including

an inability to alter allocated resources without disrupting the service, and limitations on the discretisation of the vGPU resources.

1.2 Cloud-Based Internet Cafes

As stated in §1.1 the fundamental research questions this thesis addresses are:

1. can cloud gaming be combined with internet cafes, to create a new system known as a cloud-based internet cafe, which alleviates the inefficiencies of both internet cafes and cloud gaming?
2. can optimisation algorithms provide methods for the efficient allocation of cloud resources in a cloud-based internet cafe, particularly given the use of GPU resources?

In order to answer the first question an outline of the architecture for a cloud-based internet cafe is presented, based on cloud gaming technology. The capabilities of cloud gaming technology within an internet cafe environment are not fully understood and require testing. In addition, the characteristic demands on a cloud-based cafe are not well specified. Testing is performed within a prototype cloud-based internet cafe environment to determine the capabilities of cloud technology in that environment. These capabilities are combined with internet cafe user surveys to build a representative data set for characteristic demands of a cloud-based internet cafe's users. Both the cafe capabilities and the representative data set are then used to develop and test new resource allocation algorithms capable of handling the unique requirements of a cloud-based internet cafe resource allocation problem. These new algorithms can be utilised to ensure that recent advances in GPU virtualisation enable cloud-based internet cafes to provide improved efficiency and flexibility over existing internet cafes and cloud gaming services.

1.2.1 Virtualisation

Virtualisation is both the key to cloud computing (hence cloud gaming) and the recent advancement in GPU technology that presents potential in the combination of cloud gaming and internet cafes. Before discussing this combination in detail, we summarise the concept of virtualisation.

Virtualisation is the act of creating and providing virtual resources [14]. In cloud computing virtual resources are created for CPU, RAM, Storage, Networking, and, more recently, GPU. These virtual resources are provided by mapping them onto physical resources. The virtual resources can then be used to provide services that users are demanding.

Cloud environments are created using server infrastructure and virtualisation. Servers are computers with large quantities of CPU, RAM, and storage. They are designed to be stacked efficiently in racks. Servers are placed in data centres which are able to provide the high electricity and cooling requirements of server stacks.

Within a cloud environment, server resources are provided to users via virtual machines. Virtual machines (VMs) are virtual computers provided using virtualised resources. These VMs appear as physical computers to the end user. However, the CPU, RAM, and storage on these VMs are virtual computing resources abstracted from a server's physical resources. Virtual resources are created and managed using hypervisors such as VMWare ESXi [15], Citrix XenServer [16], Microsoft Hyper-V [17], and Linux KVM [18]. These hypervisors enable a server's resources to be divided over many VMs. Users connect to the VMs using communication protocols including remote desktop protocol (RDP) and secure shell (SSH).

1.2.2 Virtual GPUs

Recently developed technology has enabled a server's GPU resource to be virtualised as a vGPU. This enables VMs on that server to share the server's GPU and provides more flexibility in terms of graphics resources for VMs. In particular, GPU virtualisation can be

used to tailor a VM's vGPU to provide the necessary gaming performance within a cloud gaming service. Previously cloud gaming services used physical GPUs with no virtualisation and the consequent limited flexibility in how the physical GPU resource is mapped to VMs.

The benefits of vGPUs for cloud gaming stems from the ability to allocate virtual resources which exactly match a user's requirements for their demanded game. This enables more efficient resource utilisation than an environment with either no virtualisation (where an entire physical machine must be allocated to a user) or no vGPUs (where an entire physical GPU must be allocated to a user). In order to fully realise the benefits of vGPU technology for cloud gaming, i.e., to make optimal use of the added flexibility, it is important to allocate resources to VMs in such a way that the physical resources are fully utilised.

The ability to allocate virtual resources, particularly vGPUs, to fully utilise the physical resources is just as important when combining the cloud gaming paradigm with the internet cafe approach. This thesis explores optimisation algorithms for solving this allocation problem (see §1.2.4). However, first the combination of cloud gaming and internet cafes needs to be described.

1.2.3 Combining Cloud Gaming and Internet Cafes

As described in §1.1, internet cafes suffer from inefficiencies in the way resources are allocated and cloud gaming suffers from high latency as a result of being provided over the internet. Alleviating high latency over the internet cannot be achieved easily, but improving the efficiency of internet cafes by using the cloud gaming paradigm is relatively straightforward.

Instead of providing individual physical resources (via a workstation) to users like a traditional internet cafe, a cloud-based internet cafe provides virtualised resources to users in the same way as cloud gaming. However, instead of providing these resources over the internet, a cloud-based internet cafe provides these resources over a local area network (LAN) within the cafe. The physical resources are provided by a server stack within a server room in the cafe and accessed by users via simple workstations with limited resources, e.g., thin client

machines or low-cost PCs. Embedding cloud gaming within internet cafes, i.e., creating a cloud-based internet cafe, provides the flexibility to allocate resources to users “on demand” and, through the use of resource allocation algorithms, enables the profit obtained from the virtualised servers to be maximised. The use of optimisation is critical for resource allocation in cloud-based internet cafe and is discussed next.

1.2.4 Optimisation

Optimisation fills an important role in cloud computing, as virtualisation allows physical server resources to be divided based on user demands. This induces an optimisation problem to determine the best way to use a cloud’s resources. This type of optimisation problem is called a resource allocation problem, and is solved differently depending on the information known about the problem and how quickly a solution is required.

Cloud resource allocation is an instance of the multi-capacity dynamic bin packing problem [19]. In this problem a set of bins, each with multiple limited capacity resources, must be packed with various objects which consume quantities of the resources. In cloud computing the objects are VMs for services which use CPU, RAM, network, and/or storage resources. Servers act as bins in the problem and are where the VMs are packed.

There are two main categories for resource allocation problems: offline and online. Offline problems (and hence algorithms for solving them – offline algorithms) have full information about the servers and VMs that must be packed. Online problems (and online algorithms) are given no future information, but have full knowledge of the servers, current VMs, and the history of arrivals and allocations to date. The extra information available to offline algorithms allow them to compute the optimal solution for any given profit function. Online algorithms obtain a percentage of the corresponding offline solution from the limited information provided by using a selection and placement strategy.

Offline algorithms are often solved using integer linear programming, or if the problem is too large, a heuristic. Linear programming is a well developed mathematical method for solving

models with requirements that can be represented by linear equations. In linear programming the model is represented with decision variables, an objective, and a series of constraints. Decision variables represent the choices being made. In the resource allocation problem these variables are the choice of server onto which a VM is placed. The objective is the total profit obtained from the choices represented by the decision variables. Constraints represent the restrictions on allocations and, in cloud computing, constraints ensure the placement of VMs do not require resources from any server that exceed that server's available resources.

More specifically an integer linear program has decision variables which must be whole numbers rather than a continuous value. In the resource allocation problem the decision variables represent the choice of server onto which a VM is placed. A partial VM can not be placed onto servers, as such 0-1 (binary) variables are used to represent if a VM is placed onto a specific server.

Online algorithms are solved using online heuristics. One type of algorithm used as online heuristics are greedy algorithms. In the cloud computing context a greedy algorithm will place VMs as demand arrives until all servers run out of resource capacity. The only decision made by the greedy algorithm in resource allocation is the choice of server onto which a VM is placed. Different choices provide different greedy algorithms and include placing VMs onto: 1) the server with the lowest resource utilisation; 2) the server with the highest resource utilisation; or 3) placing the VMs onto random servers. The greedy algorithm that performs best will depend on the properties of the problem being solved. A greedy algorithm could also include more specific placement strategies unique to different cloud services.

Other online algorithms may include rejection of users under certain conditions. These types of algorithms tend to be problem specific. A competitive algorithm is one example of an online algorithm with rejection. This type of algorithm focuses on worst case analysis. Worst case analysis considers a scenario where an adversary is sending users to the algorithm in a way which causes the algorithm to gain the minimum profit when compared to the offline solution. This research considers what is known as the strong adversary, who knows every

choice the algorithm will make including the result of random outcomes. For adversarial analysis it is necessary to prove the worst case profit of the algorithm will always be at least a ratio of the offline optimal value. This ratio is known as the competitive ratio. A competitive algorithm is one which attempts to maximise this ratio, often by balancing accepting and rejecting users.

Both offline and online algorithms for resource allocation have been applied in cloud computing for placing VMs onto servers when considering the CPU, RAM, network, and storage resources. However few algorithms consider GPU resources and fewer still the virtual GPU resources offered by Nvidia GRID [20] and AMD FirePro [21]. Those algorithms that do consider vGPUs only consider single servers rather than full cloud environments. This research gap necessitates the development of new algorithms to enable the efficient use of resources in a cloud gaming environment and, hence, a cloud-based internet cafe.

1.2.5 Using Optimisation in Cloud-based Internet Cafes

Given: 1) the inherent inefficiency of internet cafes; 2) the flexibility provided by cloud gaming, and 3) the effectiveness of optimisation to allocate resources efficiently; an internet cafe that utilises cloud gaming and optimisation together provides a powerful new paradigm for providing gaming. In a cloud-based internet cafe customers will arrive and pick a seat in the internet cafe as in a standard internet cafe. They will then select and pay per hour for a service level of their choice with each level providing access to different games, increasing in cost for higher requirement games. A VM is then created with resources allocated in a local cloud (i.e., virtualised server stack), including vGPU resources, and that VM is streamed to the customer using a remote protocol. They can select and play games as usual.

In a cloud-based internet cafe, the cloud back-end will have less total computing resources than having high performance workstations in every seat (as in a traditional internet cafe). However, unlike a traditional internet cafe, a cloud-based internet cafe will fully utilise the cloud back-end's resources that the cafe's owners have purchased. A resource allocation

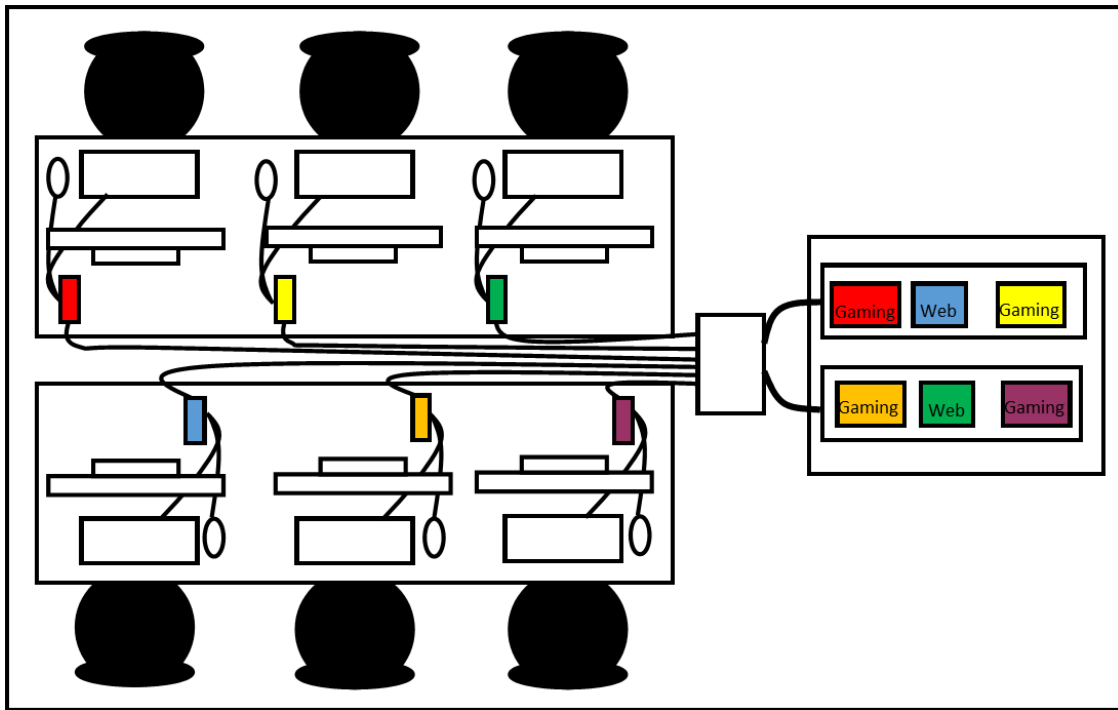


FIGURE 1.2: Example cloud-based internet cafe showing seating flexibility

algorithm will ensure that resources are used efficiently from the cloud back-end servers. This is important as lower total computing resources means that resources must be utilised efficiently to supply the same number of users as in a traditional internet cafe.

Consider the zoned internet cafe previously shown in figure 1.1. Recall that there were a fixed number of seats allocated to each zone so that resources were often under utilised or higher profit users were turned away. In the cloud-based internet cafe shown in 1.2 any seat can provide any demanded service, so the size of the zones can dynamically change as demand changes. In figure 1.2 the set-up is similar to the traditional zoned setup show in figure 1.1 except that, instead of PCs connected at each seat, each seat has a thin client which provides a connection to a VM hosted on a server. Consequently the number of Zone 1 and Zone 2 seats (using the naming convention from figure 1.1) can be adapted as needed. In figure 1.2 there are two low power, Zone 1 VMs being provided and four high power, Zone 2 VMs being provided.

Realising an optimisation-driven, cloud-based internet cafe requires analysis of and decisions on: 1) current hardware; 2) virtualisation platforms; and 3) remote desktop solutions for

cloud gaming applications. In addition, the optimisation problem must be defined. This definition includes the creation of both model parameters and a representative data set of the cafe for both servers and users. Given the problem definition, offline and online algorithms for resource allocation can be developed. The online models are useful as an operational solution for the cloud-based internet cafe and the offline models are useful for evaluating the potential of the online algorithms.

In addition to gaming, the cloud-based internet cafe has additional applications for businesses, and universities. While most staff/students only require word processing, emails, and web browsing, there are often a few staff/students that require more graphically intensive applications including Computer Aided Design (CAD) and computational algorithms such as computational fluid dynamics. Even within these applications there is variation in the demand pattern exhibited by different staff/students. It is not cost effective to have all the desktops in a workplace/university able to supply the highest intensity task even if all workers sometimes require these resources. Often workplaces have specific desktops with the required capabilities and universities have specific computer labs with specialised machines. Using the same paradigm as described for optimisation-driven cloud-based internet cafes, businesses and universities would be able to have any desk or lab access high performance resources simply by connecting to the correct VMs on a virtualised server stack within a local data centre.

1.3 Thesis Outline

This thesis presents research into the architecture and optimised operation of a cloud-based internet cafe. However, the system described throughout this thesis can also be applied within local clouds that provide graphics-intensive applications and for virtual workstations in business. This addresses the limitations of a traditional internet cafe and has the ability to offer improved flexibility and efficiency with less resources going under-utilised.

Chapter 2 summarises the current research in internet cafes, virtualisation, and resource allocation. The Resource allocation is considered both in general and in a cloud environment.

Chapter 3 presents an outline of the hardware and software options for developing a cloud-based internet cafe. Using this outline a test setup is built, benchmarked and utilised for algorithm testing.

Chapter 4 presents the mathematical description of the resource allocation problem faced in a cloud-based internet cafe and also presents the test data sets for benchmarking algorithms.

Chapters 5, 6, and 7 present algorithms which solve the resource allocation problem for a cloud-based internet cafe. Chapter 5 develops integer programs for solving the resource allocation problem to obtain optimal profits given perfect information. Chapter 6 utilises the integer programs from chapter 5 to develop a prebooking system for an internet cafe where users request seats in the internet cafe in advance. Finally chapter 7 develops online algorithms which solve the resource allocation problem with no future information. All of these algorithms are tested on data sets described in chapter 4.

Chapter 8 compares the algorithms in chapters 5, 6, and 7 and discusses their strengths and weaknesses. This chapter also considers the potential applications for the algorithms and the overall cloud-based internet cafe system.

1.4 Contribution

This thesis proposes a novel approach to improving an internet cafe's resource efficiency through the use of virtualisation in particular virtual GPUs. In order to realise this efficiency, new resource allocation algorithms must be developed which take an internet cafe's user demands and meets those demands via virtualised resources and server virtualisation. The cloud-based internet cafe system also makes use of recently developed technology in GPU virtualisation to match the available graphics resources to the demanded graphics requirements. The use of virtualised resources, in particular vGPUs, reduces the cost inefficiency

that results from leasing individual workstations with more resources than most users require (as occurs in traditional internet cafes).

Currently there is no literature describing the overall performance and capabilities of new GPU virtualisation technology from both a hardware and software perspective. We present an analysis of available technologies and benchmark the hardware performance in a gaming setting. In addition to this analysis, realistic test sets were created that describe an internet cafe's demand profiles and the servers that could be available within a cloud-based internet cafe.

Finally, this thesis presents resource allocation algorithms for efficiently meeting user resource demands by allocating resources from servers using virtualisation. Algorithms are built to operate with perfect information, partial information, and no information about future user demands. The algorithms are developed for a cloud-based internet cafe, but can also be applied to cloud gaming. However, the algorithms are less effective for cloud gaming environments due to the added affect of latency.

Chapter 2

Background

This chapter presents background concepts and previous research on internet cafes, virtualisation, cloud computing, cloud gaming, and optimisation for cloud computing. The background of optimisation for cloud computing discussed in this chapter includes both resource allocation for cloud computing and a more general optimisation model, dynamic bin packing, that can be specialised to model resource allocation. This chapter also includes a literature review of resource allocation for VMs in both cloud computing and cloud gaming.

Current research into internet cafes, presented in §2.1, consists predominantly of surveys, which provide insight into the way internet cafes are managed, and the demands of users. In particular, this research is useful for building a data set that characterises internet cafe users and the services they typically demand.

The background concepts of cloud computing, virtualisation, and cloud gaming make up the “building blocks” of a cloud-based internet cafe, and the background research describes the current state-of-the-art for building these environments. The discussion on virtualisation is limited to research that applies to private clouds, public clouds, and cloud gaming. Specific background concepts and research are presented for virtual desktop infrastructure (VDI) and vGPUs for cloud gaming. All of these background concepts and research are presented in §2.2.

Section 2.3 discusses the placement of VMs onto cloud infrastructure via resource allocation algorithms. Background concepts for resource allocation are presented along with the mathematical modelling techniques behind resource allocation algorithms. The resource allocation problem is described as a mathematical model, and the solution techniques available for solving the resource allocation problem are presented.

Finally, §2.4 presents a literature review on current solutions to the resource allocation problem for VM placement in both cloud computing and cloud gaming.

2.1 Internet Cafes

Internet cafe research focuses on the operational architecture of the internet cafe; and the demand profile and demographics of internet cafe users.

An internet cafe owner leases PCs from a supplier and also has a contract for game licenses for the computers in the internet cafe [22]. Alongside the licensed games, an internet cafe provides access to additional services including web browsing, email, and a video player. When entering an internet cafe, there is an expectation that a customer may use any seat in the internet cafe for their desired service. Owners usually charge per hour for the use of a PC. Understanding how users demand and utilise services is essential for developing a cloud-based internet cafe, where decisions must be made to utilise demanded resources efficiently.

To the best of the author's knowledge, no research has been published that develops strategies for efficiently operating an internet cafe. It is clear that internet cafes perform some minor optimisation, but any resulting findings or strategies are not published or potentially not even fully realised. One such strategy is the implementation of zoning in China which has been implemented to efficiently operate an internet cafe that has two clear groups of users: gamers and non-gamers. While gamers will use the internet cafe to play a variety of games, non-gamers use the internet cafe for checking emails, browsing the web, and watching videos. These non-gaming users do not require the computational resources that gaming users need,

so some internet cafes save money by leasing cheaper, low-performance computers to supply non-gaming users. No research has been published on the effectiveness of these zones.

No data has been released on the exact demands and profiles of internet cafe customers. This means no raw data is available on: when customers visit cafes; how long customers stay; and what services customers utilise. Fortunately, extensive research is available in the form of surveys of internet cafes both from an ownership and customer perspective. These surveys are summarised next and can provide less precise answers than raw data as to the when, how, and what questions on customers in internet cafes.

Surveys of owners include studies on the impact, growth, and drive in developing internet cafes. One survey of internet cafe entrepreneurs in Indonesia [23] investigated the factors that make internet cafes successful. Significant factors in success were regular or repeat customers, high-speed internet connections, and offering computer games as a service. Another survey on African internet cafes [24] found that internet cafes can provide affordable access to the internet and that using an internet cafe model (without charging for the service) through libraries and schools can offer improved education outcomes. Another survey in China studied the quantity and size of internet cafes in China [1] finding there to be over 185,000 internet cafes averaging 120 seats, with large internet cafes having over 400 seats.

Surveys of internet cafe users have been conducted all over the world including Turkey [25, 26], Pakistan [27], Ghana [5, 28], India [29], Malaysia [4], Uganda [30], China [31, 32], Nigeria [33], Norway [2], and elsewhere [3, 34, 35]. These surveys study the who, what, when, where, and why of internet cafe users. Some research papers survey specific subsets of internet cafe users: medical students [29]; students [33]; or adolescents [26]. Surveys of all internet cafe users are needed to create representative datasets for testing. These surveys study the demographics of internet cafe users and, most importantly: the time of day users frequent internet cafes; what services they utilise in the internet cafe; and how long users stay on average. Other demographics include gender, age, and occupation. These surveys report that most users go

to internet cafes in the afternoon or evening with around half the users playing games, and most users staying 2 to 3 hours [4, 25].

The only other noteworthy area of research on internet cafes is in the safety space. One aspect of safety is digital privacy which considers the vulnerability of internet cafes to scams and key-loggers [36, 37]. The other major safety consideration in internet cafes is physical safety, especially hygiene in internet cafes. Studies into hygiene include the number of bacteria on keyboards [38] and in the air [39].

This overall lack of research may be due to a lack of significant monetary incentive. The cost of the physical desktops cannot be significantly reduced beyond zoning without using recently developed virtualisation technology. Even zoning imposes a significant inconvenience on the customer as a customer can no longer sit in any seat. Virtualisation allows closer matching of demanded resources to supplied resources than physical computing. This virtualisation necessitates the need for research into user demand profiles and supply side optimisation.

2.2 Virtualisation

Virtualisation is the act of taking physical resources and mapping them to virtual resources [14]. Virtualisation can be performed by a private business or resources can be rented pre-virtualised from a public provider. These virtual resources must then have a virtual machine (VM) set up to use them, and that VM is connected to remotely using virtual desktop infrastructure (VDI). Cloud gaming makes use of virtualisation methods to provide games to users. More recently the advent of vGPU technology has enabled cloud gaming providers to add virtualised graphics capabilities to VMs.

This section presents background concepts and research on the use of virtualisation to: 1) build a cloud computing system from basic computing infrastructure – §2.2.1; and 2) deploy cloud services (using an existing public cloud computing system) for use by end users – §2.2.2. This cloud computing system is then accessed by users using VDI which is discussed in §2.2.3.

Finally, §2.2.4 summarises the use of **vGPU** technology in cloud gaming to enable games to be played within a cloud computing system.

2.2.1 Building a Cloud

In private clouds, server infrastructure is managed internally and, as such, systems must be built from the ground up to support virtualisation if that is the desired feature [40]. In private cloud systems, this requires a hypervisor capable of taking physical resources and converting them efficiently into virtual resources for **VMs** [14]. The major hypervisors available are VMWare ESXi [15], Microsoft Hyper-V [17], Citrix XenServer [41], and Linux KVM [18]. All these hypervisors, except for Hyper-V, are based on the Linux Kernel and, as such, support a wide variety of Linux console commands. All hypervisors are capable of creating **VMs** with virtual **CPUs** (**vCPUs**), virtual **RAM** (**vRAM**), virtual disks (**vDisks**), virtual switches (**vSwitches**) and **vGPU**.

After hypervisors are installed on a server, they boot and run in the same way that an operating system starts up. Once started a user can configure the server (including networking) and then create **vDisks** and **VMs**. Individual hypervisors are often managed by software which works remotely over multiple servers with the major management solutions being VMWare vSphere [42] for VMWare ESXi, Citrix XenCenter [16] for XenServer, Microsoft Management Console [43] for Microsoft Hyper-V, Virtual Machine Manager (or virt-manager) [44] for KVM, and OpenStack [45] for managing any hypervisor. Management software simplifies the process of creating and setting up **VMs** by providing a graphical user interface (**GUI**) solution rather than requiring the use of the command line.

Virtual resources are created either directly on a hypervisor or through management software. The process to create a virtualised resource is unique depending on the resource being virtualised [14]. The process of creating a virtual **CPU** (**vCPU**) resource is achieved by creating a **vCPU** core whose requests are forwarded on to a physical **CPU** core [14]. **RAM** is virtualised so that each piece of **vRAM** used is associated with a specific piece of physical **RAM**. Unused

vRAM may have no physical **RAM** associated until it is in use. Networking is virtualised by creating a virtual switch (**vSwitch**), which has an input port for each **VM**'s network connection and outputs for each physical network connection on the server on the target network [46]. The details of this process are abstracted when using management software.

Management software makes it possible to migrate **VMs** between servers (unless they have **vGPUs** as described in §2.2.5). This migration, at its simplest, is achieved while the virtual machine is offline. Offline migration simply requires the movement of the **vDisks** to the destination server (also known as the target server) [47]. Migration can also be performed live in which case physical resources (**CPU**, **RAM**, and networking) must be allocated on the target server for the **VM**. The **vDisk** is then copied and the **vRAM** synchronised. The user is then switched to the new **VM**, and the original **VM** is shutdown [48]. This process takes time and is network intensive, so previous research has investigated when to migrate **VMs** and which **VMs** to migrate [49, 50].

VM placement and migration is simplified by using a network storage device shared between multiple servers. Network storage devices are file stores which can be accessed as a local disk but are in fact hosted on remote servers [51]. Network storage may consist of a single server with multiple storage devices or multiple storage servers. Files are placed onto network storage according to a variety of strategies. Most recently one of the preferred strategies uses block storage and replications [52]. In this strategy, files are split into small blocks, and then a number of replications (often 3) of each block are placed onto different storage devices in a way which minimises the possibility of all replications going offline at once [52]. Examples of network storage software include Microsoft Windows Storage Server [53] and Ceph [52]. Management software (for virtualised resources) significantly simplifies the usage of network storage devices for virtualisation by providing software defined links allowing these devices to be used as local storage on **VMs**.

Both VMWare and Microsoft provide the most straightforward virtualisation solutions for building private clouds with straightforward setup and management. These solutions contain

many hidden advanced features that come with an associated high cost of a subscription to the given solution. Alternatively, Citrix and Linux provide their base virtualisation solutions for free with the option to access additional features by purchasing a subscription. To date, virtualisation solutions provide only the most basic [vGPU](#) features for free with other [vGPU](#) features available via subscription.

2.2.2 Supplying Cloud Services

Public clouds offer computing resources for lease on a pay-per-resource-per-time-unit model. Public cloud providers such as Amazon Web Services [10], Microsoft Azure [11], and Google Cloud [54] form the majority of the public cloud market. These public clouds utilise similar virtualisation technology to the private cloud technology explained in §2.2.1.

By utilising these public clouds users can complete a variety of tasks such as providing: hosted websites; email servers; virtual desktops; and high-performance computing. Public clouds can offer the same services as a private cloud, but the cloud provider manages the back-end and sells the resources to a wide variety of users [55]. Those users can utilise the public cloud services, including virtualised resources, to develop and provide their cloud-based services such as cloud gaming.

To provide a reliable service with acceptable performance around the world, public cloud providers must have servers in datacenters in many locations around the world. Each location is known as a node. The overall network of all nodes and their connections is known as a content delivery network ([CDN](#)) [56]. These [CDNs](#) must have high-speed connections between nodes with the ability to rapidly synchronise data between nodes [57]. This synchronisation is especially important for websites so updates can be quickly propagated, and also allows a user to travel around the world and maintain access to their virtual desktops.

2.2.3 Virtual Desktop Infrastructure

Virtual desktop infrastructure (VDI) provides a layer of management and access above the aforementioned management software [58]. VDI offers the ability to manage multiple sets and types of VMs, possibly being managed by multiple management software instances. A VDI also provides methods for a user to remotely connect to specific VMs using their login(s).

The major VDI solutions are Citrix XenDesktop and XenApp [59], VMWare Horizon [60], and Microsoft Remote Desktop and Remote Desktop Services [61]. XenDesktop, Horizon, and Remote Desktop offer management and remote connection protocols for using VMs as remote workstations. XenApp and Remote Desktop Services offer management and remote connection to a specific application running on a VM. VDIs are often used to supply remote desktops within a business, particularly for staff working remotely. A VDI offers security alongside high-performance computing without the need for complex software on staff laptops and desktops.

All VDIs offer at least limited support for graphical acceleration with support for OpenGL [62] and DirectX [63] rendering. This allows the utilisation of GPUs and vGPUs in a limited capacity. Microsoft's offerings fully support gaming through RemoteFX [64]. Citrix XenDesktop can run games using HDX but does not offer direct mouse capture which is required for gaming.

2.2.4 Cloud Gaming

A cloud gaming system offers the ability to play video games through a cloud environment [65]. This cloud environment may be a private cloud computing system – see §2.2.1 – or a cloud service enabled by public clouds – see §2.2.2. Clouds (private or public) provide the virtualised resources (including vGPUs) for cloud gaming systems and cloud gaming users connect to cloud gaming resources via a WAN, i.e., the internet. However, many cloud

gaming systems utilise proprietary software, and the companies have chosen not to publish their algorithms [66].

Current major cloud gaming providers are Nvidia GeForce Now [8], Sony Playstation Now [7], and LiquidSky [9]. Playstation Now utilises technology acquired from the purchase of OnLive and Gaikai [7]. This technology is based on using an entire GPU for gaming with no graphics virtualisation [67]. In contrast, GeForce Now[68] and LiquidSky[9] both make use of vGPU technology. However none of these providers have published specific details on how their services utilise GPU/vGPU resources.

Cloud gaming suffers network issues introduced by the WAN connecting cloud gaming users to cloud gaming resources. The term “network issues” broadly encompasses the overall delay between inputting a command and seeing the result of that input, as well as the quality of the image received. Natural delay of an uncongested network is difficult to improve and in fact cannot exceed, at the very best, the speed of light. Hence, current research into cloud gaming is heavily focused on reducing the congestion in a cloud gaming network and decreasing the distance image data must travel in the cloud gaming network. This research includes work on video encoding systems to reduce the network traffic caused by cloud gaming and reducing the cloud gaming network latency [65, 66]. In the rest of this subsection the research into, first, reducing network traffic and, next, reducing latency is summarised.

As a network gets congested, it is best to reduce the amount traffic being sent through the network. In cloud gaming, the majority of network traffic is generated by sending a rendered video from the cloud gaming resources through the connection to the cloud gaming user’s virtual desktop. Hence, reducing the size of video being sent enables reduced network traffic, and less network congestion, i.e., the network will work more efficiently [69]. Reducing the video’s size without negatively impacting image quality adds additional strain on the video encoding (i.e., more workload on the CPU in both the cloud gaming system and the user’s PC) and requires an intelligent compression algorithm [70]. The additional CPU workload either requires more resources at each end or more time for compression/decompression (introducing

a different source of latency). Research has found that gaming users find latencies over 100ms unplayable. This limit combined with the high framerate required for video games creates a difficult problem [71]. Gaming Anywhere is an open source cloud gaming package, which attempts to address this issue [67]. First, frames are captured from either the desktop or directly from inside the video game engine at the desired framerate. Then, various encoding and compression libraries can be used to create compressed data and send it over the network [72].

Given the aforementioned 100ms maximum latency for gaming, it is only possible to provide users with a playable experience over the internet if they are within several hundred kilometres of the servers that provide the virtualised resources [66]. This distance limitation means that for a GaaS provider to achieve reasonable service, it would need to have servers in almost every city in which that service is to be offered. Placing servers in that many cities is a costly investment as servers must be purchased and rack space for those servers must be rented in a data centre. This high upfront investment cost has stopped most cloud gaming platforms from seeing mass-market success. While it would be possible to fill some holes in such a system using public cloud resources, the typical use case for most public clouds has less stringent requirements on latency and as such the cloud CDNs do not have enough nodes to supply all locations with sufficient service. Furthermore, most cloud providers do not currently have vGPU technology installed onto servers, a requirement for gaming.

2.2.5 Virtual GPUs (vGPUs)

In order to support GPU-accelerated high performance computing and GaaS, Nvidia and AMD developed unique server GPU cards: GRID cards [20] and FirePro cards [21] respectively. These GPU cards are specifically designed for virtualisation and VDI applications. These GPU cards place multiple GPU cores onto a single card and allow each core to be shared amongst multiple VMs as vGPUs (one vGPU for each VM). Current GPU virtualisation is hardware based and related directly to the GPU card. As such, it is not possible to

live migrate a VM with a vGPU even if the destination server has the same GPU card. This limitation means the only way to migrate a VM with a vGPU is offline by shutting the VM down and removing its vGPU.

On Nvidia GRID cards, when a vGPU is created, it is allocated a core and vGPU type. Depending on the type, the core's resources will be divided into a number of equally sized parts, either: eight, four, two or one. Once this division has been performed, all VMs must be removed from the core before it can be altered [20].

Both the GRID and FirePro cards forgo external display ports (HDMI, DVI, VGA, Display-Port) in favour of special video encoding chips which output encoded video ready to be sent over a network for remote desktop. These chips limit the output framerate to 60FPS by default [20].

Research conducted by Zhang et al. looks into the scheduling of GPU cores to improve the performance of games on VMs using GPU virtualisation [73, 74]. Their research investigates the optimal way to queue GPU processing time such that game performance is unaffected. While research into improving the performance of individual cards is useful for future efficiency it is not relevant to modelling the cloud-based internet cafe problem.

2.3 Resource Allocation in Cloud Computing

The problem of how to allocate virtualised resources to users in a cloud gaming system can be modelled as a resource allocation problem in which users request and must then be allocated server resources [75]. This problem exists in two contexts: offline and online. In the offline context, all information is known a priori to the problem being solved. In the online context, only the information of users that have already arrived is known a priori to the problem being solved. This information updates over time as more users arrive and the problem consequently needs to be re-solved. The field of resource allocation in both the offline and online contexts is well studied, and that previous research is summarised next.

Resource allocation is a sub-category of the dynamic bin packing problem which is a specialisation of the bin packing problem. The classical bin packing problem takes a list $L = (p_1, \dots, p_n)$ of items and an infinite number of bins with capacity C . The size of an item p_i is given by a value $s_i \equiv s(p_i)$ where $0 < s_i \leq C$, $1 \leq i \leq n$. The objective of the problem is to pack the items into a minimum number of bins subject to the constraint that the total size of the items in any bin is not greater than C [76].

One extension of the classical bin packing problem is when the bins may also have varying capacities C_j , in which case the problem is known as a heterogeneous bin packing problem [76]. The total number of bins may also be limited such that $0 < j \leq m$ where m is the total number of bins available. This problem is known as the multiple knapsack problem (heterogeneous multiple knapsack problem if bin capacities vary) and the objective typically changes to maximise the total number of items placed into bins [77].

The dynamic bin packing problem extends the classical bin packing problem by adding a time component to the items. Each of the items is allowed to arrive and leave at arbitrary points in time. In dynamic bin packing the items have two additional properties: an arrival time $a_i \equiv a(p_i)$ and a departure time $d_i \equiv d(p_i)$ where $d_i - a_i > 0$ [78]. A further extension of the classical bin packing problem is the multi-capacity bin packing problem where each bin and item have multiple capacities and sizes. In this version of the problem, both C and s_i are vectors of multiple capacities and sizes [79]. If the vector for each bin has different capacity vectors C_j , then the problem is known as multi-capacity heterogeneous bin packing problem.

The resource allocation problem for VM placement at its most generic is a dynamic multi-capacity heterogeneous multiple knapsack problem [77, 79]. This is because:

- The items have arrival and departure times – dynamic;
- The items and bins have vectors of capacities and sizes (related to the virtualised resources) – multi-capacity;
- The bins (servers) have different vectors of capacities – heterogeneous;

- There are a limited number of bins – multiple knapsack.

This problem is NP-hard, so it is difficult to solve to optimality for a reasonable size problem. However, by making assumptions that depend on each problem instance's data, it is possible to remove the NP-hard limitation.

In a resource allocation problem, the size vector of the items and capacity vector of the bins represent demands on and availability of a variety of resources. One example of a resource allocation problem is the allocation of traffic through a broadband network [80]. In this problem, bin capacities represent bandwidth capacities of lines and switches. The items are data packets that require bandwidth from lines and switches as they move through the network over time, hence the dynamic nature of the problem. The main area of study for ICT resource allocation is in cloud and grid computing where bin capacities represent a server's physical CPU, RAM, storage, and networking; and items are users' demand for services which maps to the required computing resources [81]. Cloud gaming resource allocation adds the GPU resource to the standard cloud computing resource allocation. This extension adds many unique complexities which are not considered in existing algorithms for cloud computing resource allocation. Before considering the cloud computing or cloud gaming resource allocation, approaches and algorithms for the offline and online generic resource allocation problem are discussed.

The offline version of the resource allocation problem is most commonly solved using mixed integer programming [82] or dynamic programming [83]. Integer programming is discussed in detail in §2.3.1. In dynamic programming, the problem is broken down into a series of simple sub-problems. Each sub-problem is solved and its solution stored. The dynamic program works through the problem examining, comparing, and combining sub-problem solutions until it reaches the best solution for the overall problem [84]. The cloud gaming resource allocation problem is not suited to the dynamic programming approach in practice as each sub-problem requires an almost full enumeration of the problem space, thus removing the computational

advantage of the sub-problems and also requiring significant storage. The dynamic programming approach to the cloud gaming resource allocation problem is not considered further in this thesis.

The online version of the resource allocation problem can be solved with a wide variety of heuristics including greedy [85], evolutionary [86], and general job scheduling heuristics [87–89]. These online solution methods are discussed in §2.3.3.

2.3.1 Integer Programming

Linear programming is a solution method used to optimise a linear objective function within a mathematical model whose requirements can be represented by linear relationships [90]. The decisions in a linear program are represented by decision variables which are subject to linear equality or inequality constraints that represent the requirements. Solving the linear program involves minimising the value of the linear objective function by setting the values of the variables within the constraints. This problem is expressed in its canonical form as:

$$\begin{aligned}
 &\min c^T x \\
 &\text{Subject to:} \\
 &Ax \leq b \\
 &x \geq 0
 \end{aligned} \tag{2.1}$$

In the formulation given in (2.1), x is a vector of problem variables to be solved for, c is a vector of costs for the variables, A is a matrix of constraint coefficients, and b is a vector of limits for each constraint. In linear programming each variable in the vector x can take on any continuous value that the constraints allow. Linear programs are solved using the dual simplex method [91].

Integer programs limit some or all of the values of x to take integer values [92]. These integer variables can be useful for binary decisions where 1 represents “yes”, and 0 represents “no”,

or sets of values where only whole numbers make sense, such as deciding the number of computers to purchase. For resource allocation problems the decisions are where to place an item. The placement choice is represented with a binary variable for each bin and item pairing: the variable takes a value of 1 if the item is placed into that bin and 0 if it is not. It is not possible to put fractions of the item in two (or more) different bins, so the variable takes only integer values.

Integer programs can be solved by partially enumerating the solution space for the associated problem in an attempt to find feasible integer solutions, and the optimal integer solution is recorded. One strategy for exploring these feasible solutions is known as branch-and-bound. In a branch-and-bound strategy, first, the problem is solved with the requirement for integrality of the variables being relaxed. The resulting linear solution objective value provides a lower bound on the problem's objective value. This is because the linear solution represents the best possible objective value when variables are allowed to be continuous so that all feasible integer solutions will have objective values greater than or equal to this objective value. Any feasible integer solution provides an objective value that forms an upper bound on the problem's objective value. Often heuristics are used to find a feasible integer solution quickly. In some cases, the initial linear solution is also a feasible integer solution (the problem is known as "naturally integer"), and branch-and-bound will stop. Otherwise, branching is the technique used to enumerate the solution space. This enumeration is accomplished by splitting the solution space so that two smaller subsets are created, and infeasible (fractional) solutions that include the current linear solution are discarded. Each smaller subset is then relaxed to be linear, solved, and the enumeration process continues to split the solution space into more and more subsets while discarding infeasible solutions along the way. As it progresses, the branch-and-bound algorithm builds a tree structure where each node represents a solution where previous branches have fixed some variables to integer values, and further branches from these nodes will fix other variables to integer values. Hence, each node further down the tree contains more integer variables, but the relaxed solution represented by that node has less of the solution space, so the objective

values are increasing the further down the tree a node is. If any node's solution is greater than the current upper bound, i.e., the optimal feasible integer solution found so far, then this node is considered fathomed, and all subsequent nodes can be pruned from the tree (i.e., not explored). In some cases a subset becomes empty, and that node's linear relaxation is infeasible. Further branching will not restore feasibility, allowing this node to be fathomed. As nodes become fathomed or provide feasible integer solutions, the lower bound increases to be the minimum objective value of any node that is neither fathomed nor integer. Branch-and-bound continues exploring until all nodes in the tree are either fathomed (bounded or infeasible) or represent feasible integer solutions. At this point, the optimal feasible solution found so far is the optimal feasible solution for the original problem. Note that the difference between the upper bound and the lower bound is known as the optimality gap and branch-and-bound can be terminated early if this gap is considered small enough, usually relative to the lower bound. For full details of these solution methods refer to [82, 90, 92].

Solving a resource allocation problem using integer programming requires: 1) an integer programming formulation for the problem – Chapter 4 provides such a formulation for the cloud gaming resource allocation problem; 2) data that provides knowledge or estimates of future data on capacities and sizes – this is discussed further in Chapter 4; 3) a language for implementing the formulation and data as a problem instance; and 4) a solver for solving the problem instance. Given the formulation and full or estimated information, the solver will return the optimal solution to the problem instance. There are many programming languages and software packages for implementing integer programming formulations and solving the resulting problem instances.

Problem formulations can be implemented in a variety of languages including the currently popular options: Python [93], C++ [94, 95], and Julia [96]. The resulting problem instances can then be solved using a number of packages: Gurobi [94], IBM CPLEX [97], COIN-OR [95] (CBC [98] for integer programs (IPs)). Gurobi and IBM CPLEX are commercial optimisers capable of solving linear, mixed integer, and quadratic programming problems. COIN-OR is

an open source optimiser capable of solving linear programs via CLP [95] (COIN-OR LP), and mixed integer programs via CBC [98] (COIN-OR Branch and Cut) [99].

Integer programs for cloud computing resource allocation attempt to place virtual machines onto a set of servers such that an objective is optimised [100, 101]. Common objective functions include: maximising the profit from VMs that are running; minimising the cost of running VMs; minimising the number of servers in use; minimising the number of VM migrations required; or maximising the performance of VMs [100–103]. These integer programs may allow VMs to be migrated between servers once the VMs have started up.

These integer programs are used to solve resource allocation problems offline, §2.3.2 focuses on techniques to for solving resource allocation problems in the online setting.

2.3.2 Competitive Analysis of Online Algorithms

The quality of solution obtained by an online algorithm is often measured using competitive analysis [104]. Competitive analysis is the process of comparing the objective obtained from an online algorithm to the optimal offline objective for the same problem instance. Denote the objective obtained by a deterministic online algorithm ALG for a problem instance σ of a maximisation problem Π by $\text{ALG}(\sigma)$ and the optimal offline objective for the same instance σ by $\text{OPT}(\sigma)$. Then ALG is considered c -competitive, for $c \geq 1$, if $\text{OPT}(\sigma) \leq c \cdot \text{ALG}(\sigma)$ for every instance σ of Π . The competitive ratio of ALG is defined as the infimum over all c such that the algorithm is c -competitive, i.e., the competitive ratio $\equiv \min_{c \geq 1} \text{ALG}$ is c -competitive. The value c is also called the competitive ratio.

This c -competitiveness provides a measure for the worst case objective value produced by an online algorithm, i.e., solutions produced by the online algorithm will, at worst, have an objective value of $\frac{\text{OPT}}{c}$ for a maximisation problem.

2.3.3 Online Algorithms and Heuristics

Extensive research has been conducted into resource allocation as an online optimisation problem. The areas with the closest relationship to the online cloud gaming resource allocation problem are online interval scheduling [105, 106] and online call admission (as surveyed in Chapter 13 of [104]).

In an online interval scheduling problem, intervals or jobs with fixed start and end times are presented to an online algorithm over time. This algorithm must then decide which intervals to accept and to which machines the intervals should be assigned. The objective is to maximise either the profit of all accepted intervals or simply the number of accepted intervals, given that only one interval can be assigned to a machine at any point in time. Once started, an interval cannot be moved. In most cases the online algorithm is allowed to abort an accepted interval during its execution, but will then lose the interval and the profit from accepting it [107–112]. Settings in which termination is not allowed are studied in [113, 114].

In online call admission problems, calls arrive over time. Each call consists of two nodes within a network that need to be connected and a required (potentially changing) bandwidth of the connection along with an associated profit. All potential nodes are contained within a network of nodes and edges with various capacities on the edges. Accepting a call requires the assignment of a path through the network that connects the two nodes. This path must have free capacity at least equal to the required bandwidth for the duration of the call. The objective is to maximise the total profit of all accepted calls.

Awerbuch et al. [115] present an online algorithm for the call admission problem with a competitive ratio of $\mathcal{O}(\log nT)$ where n is the number of nodes in the network, and T is the maximum call duration. Note that this results requires the assumption that the bandwidth required by any one call is only a small fraction of the smallest edge capacity and that the profit of the call is proportional to the product of the bandwidth and duration. This online algorithm assigns each edge a cost function that is exponential with respect to the current load

of the edge and can work with multiple copies of the network. The algorithm has previously been used for several other offline and online resource allocation problems [116–120].

Further related research has been conducted on online job scheduling in cloud and grid computing environments [121]. This research considers the arrival of jobs that are then passed to a scheduler. The scheduler uses online algorithms to allocate jobs to servers that have appropriate resources as the servers become available. The algorithms in the research consider how to place jobs with respect to the requirements of other jobs in the queue to shorten the total wait time of all jobs. This consideration may lead to long waiting times for some jobs [122]. Jobs are requested over the internet into the environment, and therefore geography can be an important factor in many scheduling algorithms [123]. It is also important to consider the scalability of solutions as many clouds have thousands of servers available. As the number of servers increases computation time for job placement can contribute a significant amount of time towards the total job processing time due to the large number of options available [124]. This problem is often addressed by scheduling jobs in parallel using multiple schedules controlled by a single meta-scheduler [121]. Scheduling multiple jobs at once is a difficult problem as any conflicting allocations must be addressed before a plan can be executed. While this sort of scalability is not a significant consideration for a cloud based internet cafe with at most hundreds of users it can be an important factor for cloud gaming problems with thousands of users.

The existing literature on online and offline algorithms in the cloud computing resource allocation problem is discussed in §2.4.

2.4 Literature Review

This section contains a review of the literature on resource allocation for the placement of VMs onto servers in both cloud computing and specifically cloud gaming. This review looks at the latest research into VM placement in cloud environments in both the offline and online

settings. Two algorithms, the multi-resource algorithm and the non-migration algorithm, are discussed in detail as they relate most closely to the cloud gaming resource allocation problem.

2.4.1 Cloud VM Placement

There are many surveys of research into VM placement in the cloud. These surveys show that an extensive variety of objectives for VM placement make it difficult to compare algorithms. This variety also makes it hard to adapt existing solution approaches from specific applications.

Xu et al. survey load balancing algorithms for VM placement [19]. Their survey categorises algorithms by their:

- environment scenario – are the VMs in either public cloud(s), private cloud, or hybrid cloud (a mix of public and private clouds);
- management system – is the management of VMs either centralised (one location makes all decisions) or distributed (decisions are made at localised nodes);
- VM resource type – do the VMs use single or multiple resources;
- VM uniformity – are the VMs grouped as homogeneous (all VMs are the same), or heterogeneous (VMs are allowed to be different);
- VM allocation type – do the algorithms allocate the VMs either online (dynamically) or offline (statically);
- optimisation strategy – is the optimisation of the VM placement performed using a heuristic, a meta-heuristic, or a hybrid approach;
- scheduling process – is the scheduling process for the VMs grouped into initial placement strategies or migration strategies; and

-
- load balancing objective – the load balancing objectives consist of:
 - load variance;
 - makespan;
 - number of overloaded hosts;
 - percent of VMs located;
 - quadratic equilibrium;
 - throughput;
 - standard deviation of connections;
 - average imbalance level;
 - capacity makespan;
 - imbalance score;
 - remaining resource standard deviation;
 - number of migrations; and
 - service level agreement (SLA) violations.

Details of these objectives can be found in [19], but the relevant objectives for cloud gaming are load variance, percent VMs located, throughput, and SLA violations.

Xu et al. found that most algorithms deal with multiple resources, allocate heterogeneous VMs dynamically, and use heuristics to optimise placement. However, they also found that most meta-heuristic approaches produce better results than standard heuristic approaches (within the limited comparison that could be made). Most algorithms are also multi-objective, they not only try to optimise a load balancing objective, but they simultaneously try to optimise profits, costs, or downtime. Xu et al. also note that due to the differences between the problems addressed in each paper it is difficult to compare the algorithms. The papers surveyed by Xu et al. in [19] demonstrate the need for algorithms that:

- work in any environment scenario;

- manage VM placement centrally;
- consider multiple resources;
- allocate heterogeneous VMs;
- allocate VMs dynamically, i.e., are online algorithms;
- perform initial placement using a heuristic; and
- perform adaptive migration of VMs using a meta-heuristic.

Mann surveys the mathematical modelling problem formulations and optimisation algorithms for provisioning VMs in a cloud environment [125]. He breaks the VM allocation problems into two groups: single datacenter and multi-IaaS. Single datacenter problems handle the placement of VMs within one datacenter whereas multi-IaaS problems address VM placement in multiple datacenters. Mann states that capturing hybrid cloud scenarios (i.e., a mix of public and private clouds) will require a convergence of the two problem types. Mann also states that the heterogeneity of problem formulations makes it challenging to compare algorithms and he notes the need for a method to compare algorithms. Mann concludes that, to select a potential algorithm for a given problem type or problem instance, one must characterise the problem type/instance into categories similar to those proposed by Xu et al. [19].

Milani et al. conduct a systematic and comprehensive review of load balancing algorithms for VMs in the cloud [126]. The review focuses on dynamic (i.e., online) and hybrid (i.e., mixed offline and online) load balancing techniques. They found hybrid algorithms have increased performance over purely dynamic algorithms. It is noted that no algorithms have been presented for cloud load balancing that can handle a variety of objectives, choosing instead to focus on a single objective or single group of objectives. Few algorithms can handle objectives beyond what they were designed for. Load balancing algorithms can improve cloud resource utilisation and increase overall cloud performance. Milani et al. conclude that algorithms

addressing load balancing are unable to handle heterogeneous systems of VMs, and have low scalability, both of which are a requirement for a real on-demand cloud system.

Mills et al. present an unbiased method for comparing placement algorithms [127]. The method compares initial VM placement algorithms in simulations of large distributed systems. They compare 18 VM placement algorithms and conclude that the selection of a suitable server group is more important than the selection of a specific node/server. The placement of a VM on specific nodes in on-demand infrastructure only makes a small difference in overall profits while the selection of a grouping has the potential for substantial, long-term profit gains.

Using the groups from these surveys key papers on VM placement in cloud computing are summarised in table 2.1. This review focuses on algorithms with applications which work without live migration. Summary paragraphs of each paper are presented in the remainder of the chapter.

TABLE 2.1: Summary of key VM placement papers

Paper	Field	Objective type	Load Objective	Algorithm Type	Information	Strategy	Migration
Gupta et al [128]	High Performance Computing	Single	Maximise throughput	Heuristic	Offline	Grouping VMs	No
Calcavecchia et al [129]	VM placement	Multi	Minimise weighted sum: number of migrations, load variance, and SLA violations	Heuristic	Online	Historical Usage	Limited Quantity
Hyser et al [130]	VM Mapping	Multi	User defined	Heuristic	Online	VM to server mapping	Yes
Chaisiri et al [100]	VM placement	Single	Minimise cost	Stochastic integer program	Hybrid	Grouping VMs	No
Fang et al [101]	VM placement	Single	Minimise power usage	Heuristic	Offline	Network planning and VM placement	No
Xu and Fortes [102]	VM placement	Multi	Maximise throughput, minimise power usage and heat produced	Heuristic	Offline	Genetic algorithm	Future Work
Chang et al [103]	VM placement	Single	Either minimise cost, maximise throughput	Heuristic	Online	Multiplicity	No
Leinberger et al [131]	VM placement	Single	Minimise load variance	Heuristic	Offline	d-capacity bin packing	No

Gupta et al. present an application-aware VM placement algorithm for high performance computing clouds [128]. They develop the VM placement algorithm in OpenStack and test it using CloudSim. The VMs are placed using a distinct set of heuristic rules. Gupta et al. find that the cloud benefits from placing VMs into smart co-locations within server infrastructure i.e., placing related VMs in such a way that no VM is slowing down the overall process. They also find that knowledge of the application running on the VM can benefit

placement algorithms. Their algorithms show a 32% improvement in job throughput and performance improvements of up to 45% using their algorithm over random VM placement. This research demonstrates the potential for intelligent placement algorithms in high-performance computing and the potential of grouping VMs for intelligent co-location on servers.

Calcavecchia et al. present a practical model for cloud placement called Backward Speculative Placement (BSP) [129]. BSP projects the past resource demands of VMs onto a candidate server. This strategy is used to place VMs given a stream of arriving requests and to periodically optimise the system to handle changing demand. Calcavecchia et al. compare BSP to a mixed integer program (with limited solve time) showing improvements in solution time from minutes to under a second with similar solution quality. Their BSP algorithm shows the potential of utilising historical VM resource demands when making placement decisions.

Hyser et al. present a virtual machine placement system which uses an autonomic controller to dynamically map VMs to physical hosts depending on user policies [130]. They show that the mapping problem is important and computationally challenging. The system presented offers transparent live migration of VMs to match new mapping strategies. The usage of a mapping strategy is significant as within a large datacenter it is challenging to optimise individual VMs or to reallocate VMs manually. Instead, system-wide mapping strategies are required for general rearrangement.

Chaisiri et al. propose an optimal virtual machine placement algorithm which uses stochastic integer programming to minimise the cost of hosting each VM in the cloud [100]. The algorithm takes a subset of VMs and places them into a cloud provider's system that charges for computing power consumed. They then consider the decision of reserving resources in advance for a lower price or purchasing resources on demand when there is uncertainty in how many VMs will be demanded. They create a stochastic integer program to optimise reservation and on-demand purchasing with uncertainty in price and VM demand. They claim that the stochastic integer program can achieve the lowest total cost possible even under uncertainty.

Fang et al. present VMPlanner, a power saving algorithm which attempts to place VMs such that as many network devices as possible can be shut down to save power [101]. VMPlanner provides a network-wide power manager which optimises VM placement and traffic flow routing to reduce data centre power costs. A combinatorial optimisation problem is formulated and decomposed into three problems: traffic aware VM grouping; distance aware VM-group to server rack mapping; and power-aware inter-VM traffic flow routing. VMPlanner integrates efficient approximation algorithms to solve these problems. Fang et al. claim VMPlanner can outperform other network power algorithms. They demonstrate that combining VM placement with network planning can potentially improve power usage in data centres. They note that VMPlanner could be further improved by including formulation on VM relationships in terms of network communication and the addition of an intelligent reallocation and migration algorithm.

Xu and Fortes present a multi-objective optimisation problem for simultaneously minimising total resource wastage, power consumption, and thermal dissipation costs that is solved using a genetic algorithm [102]. This specialised genetic algorithm is proposed to efficiently deal with the large solution space in problem instances for large-scale data centres. Their initial placement algorithm shows superior performance compared to standard bin packing algorithms and single objective optimisation. They also note that VM migration adds additional flexibility but is a costly activity and propose an additional objective to handle VM migration opportunities which minimises the impact of VM migration.

Chang et al. formulate the VM placement problem as a resource allocation problem with multiplicity [103]. Multiplicity is the concept of allocating a quantity of resources for a specific task where a number of users are estimated to demand that task. This is achieved by allocating resources in groups of tasks. For example, resources may be allocated to run virtual private networks (VPNs) for an estimated 4 VPN users. These groups of resources are utilised in a similar way to resources in the interval scheduling problem in which VMs are placed and use resources within the allocation for an interval before releasing the resources for another user. These resource demands are allowed to be heterogeneous over the different

resource types. This heterogeneity is simplified into five resource demands for each task. An approximation algorithm to reduce costs or improve resource utilisation is developed using this problem structure, which applies to both private and public clouds.

Leinberger et al. present two d -capacity bin-packing algorithms for scheduling jobs in a parallel processing system [131]. They develop permutation pack and choose pack algorithms which place items with improved scaling and the ability to account for multiple capacities. Permutation pack attempts to place items into bins by ordering placement by the bin capacity to item size ratio. The result is each capacity within a bin is almost equally loaded. Choose pack instead groups items into large and small size groups for different capacities. Choose pack then attempts to pack these groups, so all capacities are roughly equally loaded within each bin. They find these algorithms can improve performance over first fit packing. This improvement shows that in situations where the capacities have little correlation in usage, a smart balancing algorithm, which considers the different capacity demands from items, can improve algorithm performance.

2.4.2 Cloud Gaming VM Placement

Cloud gaming VM placement is a less well studied field with few existing algorithms for VM placement in a data centre with GPU virtualisation.

Cai et al. survey current research into cloud gaming platforms and the optimisation of those platforms [132]. They present cloud gaming platforms which support quantitative performance measurements, including quality of service and quality of experience measures. They also present optimisation research which is split into two major categories: optimising cloud server infrastructure, including resource allocation and distributed architecture; and optimising communication, including data compression and transmission. Research into resource allocation in cloud gaming has been conducted on both VM placement and cloud scheduling to improve the quality of cloud gaming services. The surveyed research into resource allocation mainly consists of intelligent GPU virtualisation algorithms and methods for maintaining

the quality of experience when sharing GPUs. However, research has also been conducted into VM placement that considers: quality of experience; minimising the number of nodes required; and predicted user end times. Cai et al. conclude that further optimisation of resource allocation is needed in cloud gaming to make services profitable.

Current research papers into VM placement in cloud gaming are summarised in table 2.2, and in the remainder of this section. Current research does not handle the full complexity of vGPUs with papers either simplifying the problem or ignoring it. The objectives are focused on SLA violations and maximising throughput.

TABLE 2.2: Summary of cloud gaming VM placement papers

Paper	Field	Objective type	Objective	Algorithm Type	Information	Strategy	Level
Li et al [75]	VM placement	Single	Maximise throughput	Heuristic	Online	Competitive greedy	VM simplified vGPU
Li et al [133]	VM placement	Single	Maximise throughput	Heuristic	Online	Neural network	VM simplified vGPU
Zhang et al [73, 74]	GPU scheduling	Single	Maximise throughput or minimise SLA violations	Heuristic	Online	GPU controller	GPU
Hong et al [134]	VM placement	Multi	Maximise profit and minimise SLA violations	Integer program and heuristic	Offline	Quality driven	VM no vGPU
Finkel et al [135]	Evaluation	Single	Maximise throughput	Heuristic	Online	Hill-climbing algorithm	Server per user

Li et al. present the MinTotal Dynamic Bin Packing problem solved using first fit, best fit, and any fit online algorithms for placing VMs in a cloud gaming system [75]. This research has demonstrated potential competitive ratios for greedy strategies for a simplified version of the cloud gaming problem. They find competitive ratios which relate to the ratio between the minimum and maximum duration any user can stay. This ratio is defined as $\mu = \frac{\max \text{duration}}{\min \text{duration}}$. They also assume all users will utilise the same quantity of resources. They find competitive ratios of $\mu + 1$ for any fit, $2\mu + 7$ for first fit, and $\mu + 5$ for hybrid first fit, with best fit being unbounded (i.e. has no limit on how badly it can perform). It is important to note this paper does not consider GPU virtualisation with its full complexities and constraints.

Li et al. also present a placement strategy for cloud gaming resources which attempts to place users based on predicted session end times [133]. This research predicts end times of users with a neural network. They find that the added information from estimated end times significantly improves resource utilisation when compared to first fit or best fit algorithms.

Due to the daily play frequency of gamers, this approach works to predict session lengths accurately and reduces wasted cloud resources. The paper assumes that GPU is the only resource that is a bottleneck for virtual machines.

Zhang et al. present VGRIS (Virtualized GPU Resource Isolation and Scheduling), and vGASA (virtualised GPU Adaptive Scheduling Algorithm) resource management frameworks for cloud gaming that seek to share GPU resources between VMs [73, 74]. VGRIS utilises agents on VMs and a central controller to regulate GPU usage from individual VMs. VGASA utilises agents on VMs and a central controller in a paravirtualisation framework to regulate GPU usage. The VGASA regulation runs with three strategies: 1) scheduling algorithms allocate enough resources to meet minimum performance requirements; 2) VMs are given GPU resources proportional to their weight; 3) a hybrid algorithm that allocates resource to meet minimum performance requirements, giving extra resources proportional to a VMs weight. All algorithms can provide game performance only a few FPS lower than native GPU. Performance is reduced by between 5% to 10% in 3DMark tests for the proportional algorithm. This research shows the potential for vGPUs to spread resources amongst multiple VMs given intelligent load balancing.

Hong et al. investigate optimal cloud gaming resource allocation to both maximise profit and maintain a high quality of service [134]. This research develops an integer program which selects and places VMs onto servers based on available resources and is solved using CPLEX. They also develop a quality driven heuristic to solve the problem efficiently. This quality driven heuristic places VMs onto a single server until the quality of experience drops below a certain level while maximising total profit. The integer program is unable to scale to large problem instances. They build an extensive test bed and run trace-driven simulations to show that the heuristics can obtain up to 90% of the profit and 100% of the quality of experience compared to the integer program while only taking a single second to solve. However, their solution does not include GPU virtualisation instead passing through full physical GPUs to VMs and is only tested for two VM types in a homogeneous system.

Finkel et al. build a simulation to evaluate the effectiveness of OnLive's resource distribution strategy [135]. OnLive dedicates entire servers to individual games with no virtualisation. OnLive attempts to predict which games will be requested and allocates hardware for those games in advance. This research builds a simulation utilising traces from OnLive's historical service demand and can build effective distributions of user demands. They find that user demand is best satisfied by using a hill-climbing algorithm.

Current research into the cloud gaming resource allocation problem has investigated placement of VMs in a resource allocation problem with limited GPU virtualisation and efficient usage of vGPUs for individual VMs. The research has not developed algorithms for resource allocation of VMs with the full complexity of GPU virtualisation and its limitations. Existing research has built greedy algorithms for limited GPU virtualisation with competitive ratios relating to user stay duration's. These competitive algorithms rely on homogeneity of server and user demands. An integer program has also been developed which solves the cloud gaming resource allocation problem for pass-through GPUs.

This thesis presents online and offline algorithms for a cloud gaming resource allocation problem for a cloud-based internet cafe. The online algorithms include greedy algorithms which are able to handle heterogeneous servers and user demands, and a competitive algorithm which enables user rejection and provides a competitive ratio for heterogeneous servers and user demands. The offline algorithms are integer programs which function similarly to those previously developed in research with the additional ability to handle the complexities of GPU virtualisation.

Chapter 3

Building a Cloud-Based Internet Cafe

Before investigating algorithms for resource allocation/[VM](#) placement in a cloud-based internet cafe, the hardware and software architecture of the cloud-based internet cafe needs to be understood. This chapter explores and analyses the potential hardware and software for creating a cloud-based internet cafe. Building a cloud-based internet cafe requires both a physical and a virtual environment. The physical environment consists of the physical servers within a server rack which will supply services to the users in the internet cafe. These servers require supporting virtual desktop infrastructure ([VDI](#)), networking and thin clients enabling users to connect to the servers. The virtual environment consists of [VMs](#) for running the services, hypervisors that virtualise the physical resources of the servers for use by the [VMs](#), and remote desktop software to connect users to their [VMs](#).

To investigate resource allocation in a cloud-based internet cafe, a prototype of the physical and virtual environment, referred to here as the “test setup”, was created. Servers were purchased to provide test hardware for the physical environment. In particular, Nvidia GRID [GPU](#)s were used to test graphics virtualisation technology. Hypervisor and remote desktop software was installed on these servers to create the virtual environment for testing.

Performance testing of the hardware setup was completed to investigate the hardware’s ability to provide games in a cloud-based internet cafe. Such an environment had not been previously implemented, so it was challenging to create a viable setup. Multiple software applications were also tested for viability in the virtual environment. In particular, it was challenging to find high-performance remote desktop software. All testing was performed in the test setup under a variety of configurations to measure the performance of a prototype cloud-based internet cafe environment. This testing provides inputs for the test data sets in §4.3.

This chapter is organised to build a cloud-based internet cafe from lowest to highest level. First presenting hardware including; servers, networking, storage, and vGPU. Followed by software including; hypervisors, virtualisation, and remote desktop. Finally, the “test setup” is performance benchmarked.

3.1 Hardware

In this section the physical environment is discussed in detail including sections on the servers, the GPUs, and the storage.

The prototype cloud-based internet cafe, i.e., the test setup, consisted of two graphics servers that are described in §3.1.1, twelve “thin” clients, two switches, and a management server. Part of the test setup (with only six thin clients) is shown in figure 3.1. For test thin clients old engineering lab machines were used. These are Dell T3500 machines, i.e., older, low-performance machines, running Ubuntu 14.04.

Two EX4200-24T Juniper switches were used to connect the servers to the thin clients (named Green and White respectively). Each server is connected to each switch with a single 10 gigabit per second fibre channel small form-factor pluggable transceiver plus (SFP+) connection. Each thin client is connected to each switch with a single CAT6 gigabit Ethernet connection. Each switch is a separate network and subnet; the “green” network and “white” network depending on which switch the network utilises. When a connection from a thin client to a

VM on a server is requested either the green or white network is selected for that connection. A separate VM managed the network settings on the management server.

The management server is connected with dual gigabit Ethernet to both switches and hosts a VM running pfSense that provides networking configurations for the cloud-based internet cafe[136]. PfSense is open source firewall/router software based on FreeBSD [136]. This VM acts as a gateway and firewall to the internet. The VM also provides a dynamic name server (DNS) for both the green and white local networks as well as for external names. In addition, the VM acts as a dynamic host configuration protocol (DHCP) server for both the green and white networks, allocating IP addresses for both thin clients and VMs created on the servers. Figure 3.1 shows the connections between all components of the setup.

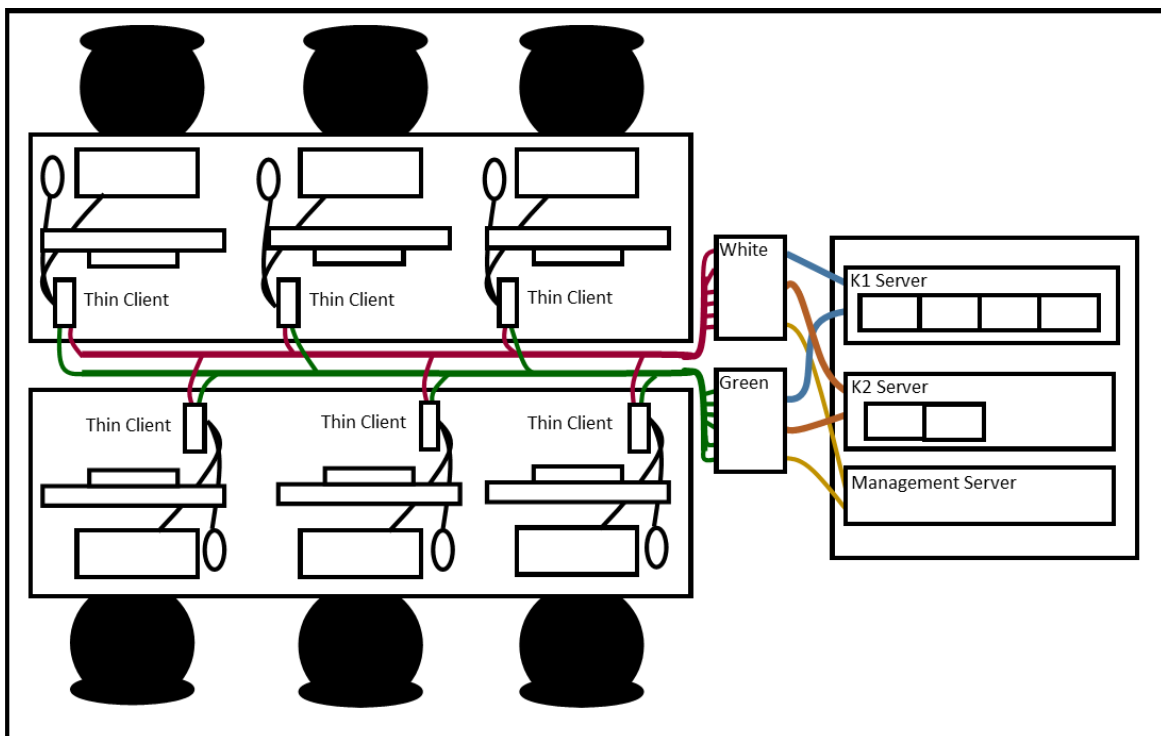


FIGURE 3.1: Test Cloud-Based Internet Cafe Setup

The red lines represent the CAT6 gigabit Ethernet network connecting the thin clients to the White switch, the green lines are the CAT6 Gigabit Ethernet network connecting the thin clients to the Green switch. The blue 10 gigabit fibre channel SFP+ network connects each switch to the K1 Server and the brown 10 gigabit per second fibre channel SFP+ network connects each switch to the K2 Server. The gold dual CAT6 gigabit Ethernet network connects each switch to the Management Server.

The management server also provides any VMs needed to control or manage features in the test setup. This management server has a 16 core Intel Xeon X7350 CPU at 2.93GHz

and 93GB RAM with no GPU. The server always runs the pfSense gateway VM and runs additional VMs as required for monitoring activity and for the controllers that any VDIs require.

The two graphics servers provide the VMs which supply users with their desired service.

3.1.1 Servers

The test servers for the cloud-based internet cafe were two Supermicro servers. These servers have high specifications to provide the VMs for the demanded services. They also have high-speed connections to the thin clients to provide stable remote connections.

These servers each have a 32 core Intel Xeon E5-2650 CPU at 2.4GHz, 65GB of RAM, a 256GB Samsung solid state drive (SSD), dual gigabit Ethernet network connections for internal management, and 10-gigabit SFP+ fibre channel networking to each switch for remote connections with the thin clients. One server has a Nvidia GRID K1 GPU card, and the other has a Nvidia GRID K2 GPU card. The GRID K1 is for workstation level graphics and the GRID K2 for high-performance tasks. Table 3.1 summarises these specifications. §3.1.2 talks about the specifics of the graphics cards. The servers also required additional network storage to adequately supply VMs to users. This network storage is discussed in §3.1.3.

TABLE 3.1: Specifications of servers used for feasibility testing

Name	CPU	Cores	Clock	RAM	GPU	Storage	Network
K1 Server	Intel Xeon E5-2650	32	2.6GHz	65.5GB	Nvidia GRID K1	256GB SSD	x2 10 gigabit Fibre Channel
K2 Server	Intel Xeon E5-2650	32	2.6GHz	65.5GB	Nvidia GRID K2	256GB SSD	x2 10 gigabit Fibre Channel

3.1.2 GPU

The GPU cards discussed here were used to supply the graphics processing required for gaming VMs. The test servers had Nvidia GRID K1 and K2 GPUs [20].

These GPUs use Nvidia Kepler cores equivalent in performance to the Nvidia 700 series cards. These cards are designed for server infrastructure with passive or end-to-end cooling. They make use of two PCIe 3.0 slots for improved bandwidth and contain a unique H.264 video encoding acceleration block in place of video output ports as they are designed for remote use only. Cards make use of Nvidia CUDA cores and support the following rendering application programming interfaces (APIs): CUDA, DirectX 9, 10, 11, and OpenGL 4.3. The specific APIs supported depend on the hypervisor used for virtualisation.

A Nvidia GRID K1 has 4 Kepler GK107 GPU cores integrated into the GPU. Each core is equivalent to a Geforce GT 720 in architecture [20]. However, the cores offer slightly higher speeds (of 850MHz) and double the DDR3 video random access memory (VRAM) with 4GB per core (at 891 MHz). The K1 layout is shown in figure 3.2. The GPU card draws 130W at peak power.

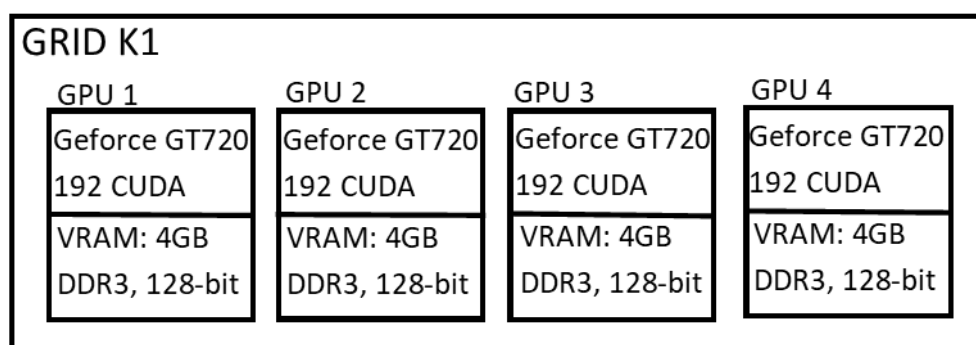


FIGURE 3.2: Nvidia GRID K1 card layout and specifications

A Nvidia GRID K2 has 2 Kepler GK104 GPU cores with each core similar to a GeForce GTX 770 [20]. The cores offer slightly lower clock speeds than the K1 cores (of 745MHz) and double the GDDR5 VRAM of K1 cores with 4GB per core (at 2.5GHz). The K2 layout is shown in figure 3.3. The board draws up to 225W of power.

The advantage of using Nvidia GRID cards over specific server augmented desktop GPU cards comes from: the additional PCIe bandwidth; the built-in video encoding hardware; and the extra performance offered from multiple cores. The combination of these factors enables

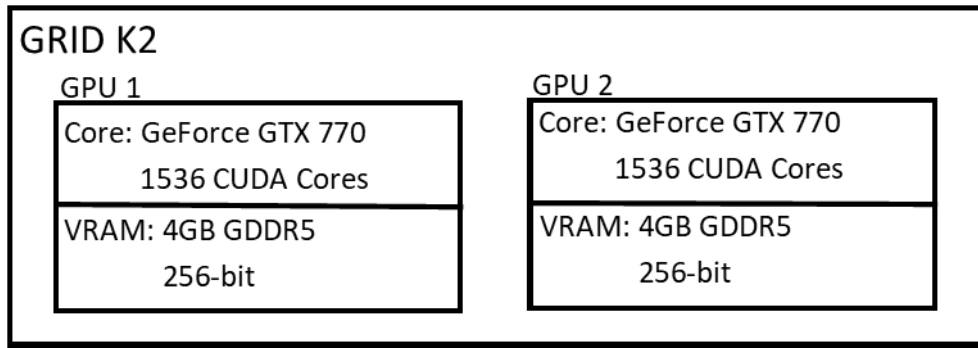


FIGURE 3.3: Nvidia GRID K2 card layout and specifications

virtualisation of the GPU cores in a similar way to how CPU and RAM can be virtualised in a traditional cloud environment.

3.1.2.1 vGPUs

The virtualisation of GPUs to create vGPUs offers cloud gaming management the ability to make meaningful decisions for resource allocation while supplying gaming users with high performance VMs.

TABLE 3.2: vGPU names and specifications

vGPU Name		Max VMs per core	VRAM (GB)
GRID K1	GRID K2		
K100	K200	8	0.25
K120	K220	8	0.5
K140	K240	4	1
K160	K260	2	2
K180	K280	1	4
Pass (K1)	Pass (K2)	No virtualisation (1)	4

GPU virtualisation, while more flexible than a standard desktop GPU, still has restrictions on how the GPU cards can be used. The virtualisation of the cards involves the creation of a vGPU for each VM. A single GPU core on the card can be virtualised into either eighths, fourths, halves, or one whole core. These vGPUs are named, in correspondence to their GPU card as either K100s (K1 card) or K200s (K2 card), with the lowest performance vGPU named K100/K200 and increasing by 20 as the resources rise until the K180/K280. These vGPUs share core time equally and take a fixed portion of the core's VRAM. These portions

are visualised in figure 3.4 and fully described in table 3.2. Shared cores process the requests from the vGPUs in a simple First-In First-Out (FIFO) queue method. This sharing means that if only one VM is on the GPU core, no matter what vGPU type is being used, that VM will receive the full core speed. If two VMs are on a GPU core, then they will receive a proportion of the core speed relative to the number of requests they send per second. As such it is possible to run a high-performance task on a low power vGPU (e.g., a K220) if it is the only VM on the core and requires a relatively small quantity of VRAM. This split is illustrated in figure 3.4 which shows the K1 and K2 cores with the top half of each core representing the undivided (shared processing) core and the bottom half the divisions of RAM between VMs.

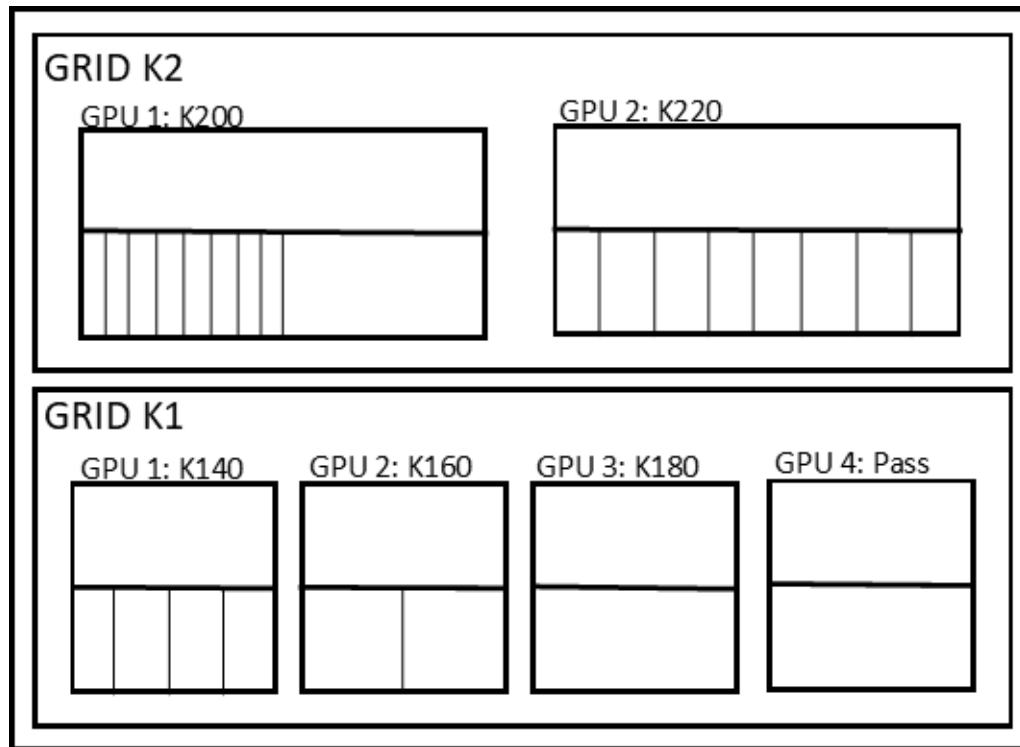


FIGURE 3.4: Types of vGPU with RAM and core divisions

It is important to note that when a VM with a vGPU is created, it is not possible to live migrate that VM to another server or GPU core. Instead, it is necessary first to shut down that VM then move its vDisk to the destination server or restart the VM with a new GPU core selected. This migration is different from a VM that is not using a vGPU which may be live migrated to another server. This limitation is because GPU virtualisation is a low-level

process which interacts closely with the hardware on which the VM is placed. For more details see [20].

In addition to not being able to migrate VMs with vGPUs it is also not possible to change the vGPU type of a VM without first shutting down that VM. Furthermore, when a VM with a specific vGPU is run on a GPU core, the entire core may only run other VMs with the same vGPU type. Of course it is possible to change the vGPU type of a VM before starting it to enable it to share a GPU core, but the VM will only meet its performance requirements if the vGPU type it is using provides greater graphics performance than its designated vGPU type. However, since: 1) cores are restricted to a single vGPU type for all VMs on the core; and 2) a VM cannot change its vGPU type without shutting down; a core will only be able to run VMs of the same, vGPU type until all VMs running on that core are shut down.

For example figures 3.5 and 3.6 show invalid and valid VMs running on GPU cores respectively.

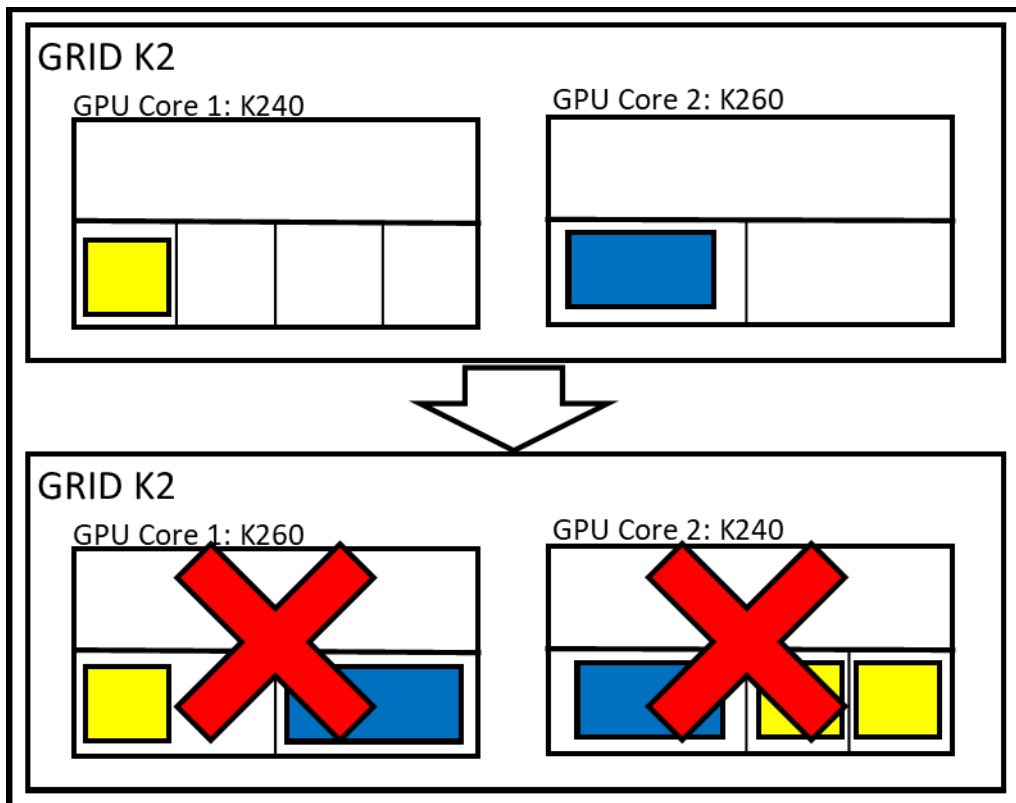


FIGURE 3.5: Invalid VM additions to GRID K2 card

Summarising the invalid configurations shown in figure 3.5, core 1 has a VM running which uses a K240 vGPU, which means the core can only run other K240 VMs. Similarly, core 2 is running a K260 vGPU. If a new VM running a K260 vGPU (referred to as a K260 VM) and two new VMs running K240 vGPUs (referred to as K240 VMs) are added, then the K260 VM cannot be placed on core 1 and the K240 VMs cannot be placed on core 2. Both placements violate the current vGPU types on the cores. However, the K240 VMs could be upgraded to use K260 vGPU and then placed on core 2. Unfortunately, the K260 vGPU type only supports a maximum of two VMs, so it is not possible to add two more VMs to core 2. Note that the K260 VM currently on core 2 could be shut down and “downgraded” to a K240 VM, and the core could then support this downgraded VM and the two new K240 VMs, but the performance of the service on the downgraded VM may be compromised.

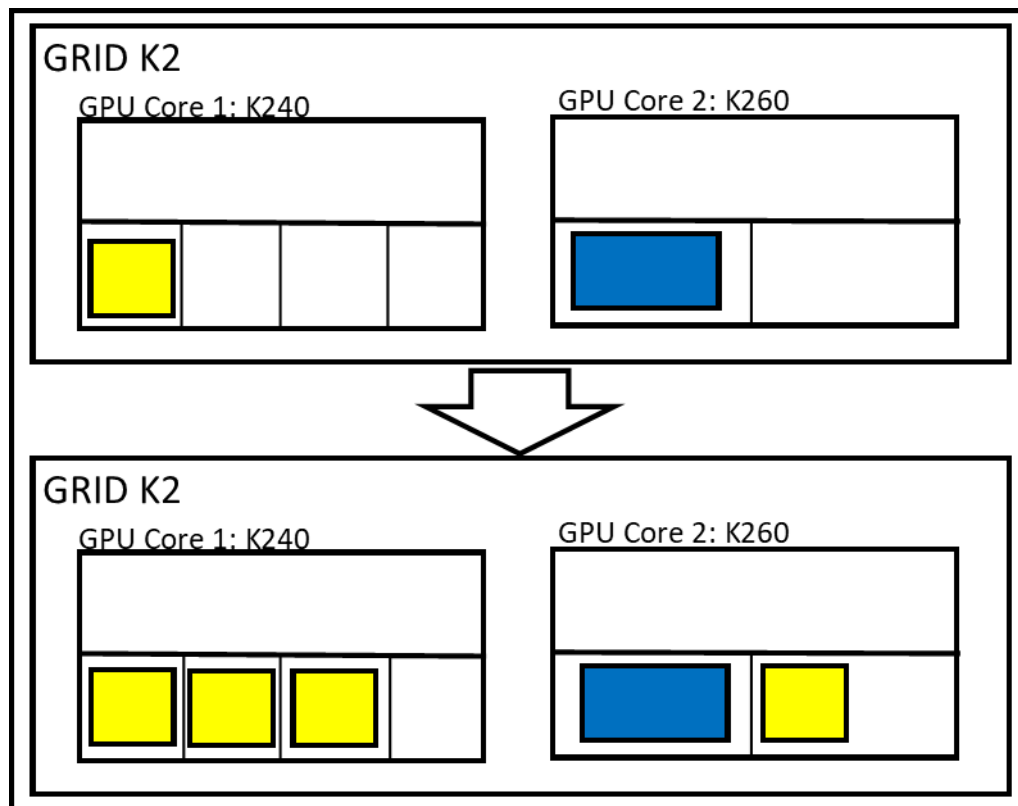


FIGURE 3.6: Valid VM additions to a GRID K2 card. Note that the K240 VM on the K260 card will have higher than specified graphics performance

Summarising the valid configurations shown in 3.6, both core 1 and core 2 start with VMs as in the invalid configurations (see figure 3.5). However, now the two new K240 VMs are

added to core 1 and a single K240 VM (running as a K260 vGPU) is added to core 2. The vGPU types are consistent and the number of supported vGPUs of each type is sufficient.

One key performance measure for a cloud-based internet cafe is the framerate of the VMs that exist within the cloud. Testing was carried out by connecting to these VMs using Steam Sharing and using Steams' built in frame per second (FPS) counter to record the performance of a variety of games averaged over a 5 minute period. This extensive testing of both the GRID K1 and K2 card resulted in the game performance levels shown in table 3.3. Note that 60 FPS is optimal for gaming, but a base rate of 30 FPS is considered playable [137]. Testing showed that vGPU performance was sub-optimal for supporting gaming and in some cases, the performance would be considered unplayable. Neither card was able to play the latest games at 1080p on high settings at 60 FPS, with the K2 able to run Witcher 3 at 30 FPS on medium settings in-game when given a K280 vGPU, and the K1 was barely able to reach 10 FPS on the lowest settings at 1080p. The K1 core was not able to exceed 20 FPS for lower requirement multiplayer games like Overwatch, or Warframe, although a K2 core was able to run two of these games using K240 vGPUs while maintaining a steady 50 FPS at 1080p with medium settings. Performance reached 60 FPS at 1080p on medium settings with a single VM on the K2 core. The K1 core was able to run a single instance of League of Legends, World of Warcraft, or DoTA2 at 40 FPS at 1080p on medium settings with a K180. The K2 core was able to run 4 copies of League of Legends, World of Warcraft, and DoTA2 with a K240 at 40 FPS at 1080p on medium settings. With only two copies of these games running the K2 core was able to reach 60 FPS at 1080p on medium settings. More in-depth analysis of overall server performance is discussed in §3.4.

3.1.3 Storage

Running tests on the cloud-based internet cafe prototype requires storage beyond the K1 and K2 servers' local 256GB SSD. This section describes the network storage used in the test setup.

TABLE 3.3: GPU game performance with various settings

	GPU	vGPU	VMs/Core	Avg FPS	Resolution	Settings
Low Requirement: League of Legends World of Warcraft	K1	K180	1	40	1080p	Medium
	K2	K240	4	40	1080p	Medium
	K2	K260	2	60	1080p	Medium
	K2	K280	1	60	1080p	High
Medium Requirement: Overwatch	K1	K180	1	20	1080p	Low
	K2	K240	2	50	1080p	Medium
	K2	K280	1	60	1080p	Medium
High Requirement: Witcher 3	K1	K180	1	10	1080p	Lowest
	K2	K280	1	30	1080p	Medium
	K2	K280	1	45	1080p	Low

Current internet cafes have local storage for the operating system and remote shared storage for the video games. When using VMs, it is also a good idea to use a storage area network (SAN) or network-attached storage (NAS) for all vDisks. A special storage server was set up with four 250GB Samsung 850 EVOs SSDs, for a total of 1TB of storage, to provide networked storage for the test setup. This storage server runs Ubuntu server 14.04 with the drives in a RAID0 array using the BTRFS file system. This drive was connected remotely using asynchronous Network File System (NFS). NFS is supported by default on XenServer (the hypervisor used for the test setup). Asynchronous access means the NFS immediately considers write requests complete while the data is still in memory, as opposed to synchronous, which doesn't consider a request complete until it has been written to the physical disk. It is important to set a NFS to asynchronous access so multiple vDisks can access the NFS simultaneously. If disk requests are forced to queue under synchronous access, then the NFS is extremely slow. This access comes at the risk of losing data if the storage server crashes while requests are still in memory.

The remainder of this section presents the software operating on the hardware presented previously. Comparing the capabilities of different software options available when run on the stated hardware in table 3.1.

3.2 Hypervisors

The hypervisor is the software installed on the servers that virtualises resources and enables the creation of VMs. A hypervisor is crucial in creating a prototype cloud-based internet cafe in which the VMs are the key to supplying services. Hypervisors include Citrix XenServer, VMWare ESXi, Linux KVM, and Microsoft Hyper-V.

The most important feature the hypervisor needs to be useful in our test setup is support for vGPUs. Without this support, the cards described in §3.1.2 cannot provide any of the flexible features of the GRID GPUs that are being investigated.

3.2.1 Citrix XenServer

Citrix XenServer was the only hypervisor supporting vGPUs at the time the servers were purchased. Additionally, Citrix is the official partner of Nvidia for their vGPU technology. Citrix also offered academic licenses for testing purposes. For these reasons, XenServer was used as the hypervisor for all phases of testing.

Citrix XenServer 5.1 was the first hypervisor to offer vGPU support and was the official release hypervisor for the Nvidia GRID cards. When the servers were first set up, it was the only hypervisor available for testing vGPU technology. Citrix XenServer 6 is the most recent version of Citrix XenServer that offers free access to vGPU functionality. Citrix XenServer 7 offers significant stability improvements for vGPUs as well as additional vGPU configurations. However, access to these vGPU settings is restricted to customers with XenServer Enterprise edition. This version was used for the most recent test setup with an academic license provided by Citrix.

XenServer in our setup is managed by XenCenter which provides an easy GUI for managing networking, storage, and for creating VMs with vGPUs.

3.2.2 Other Hypervisors

Both VMware ESXi and Linux KVM (via Redhat) now support GPU virtualisation in some form. However, neither of these hypervisors were investigated in the test setup. While VMware ESXi offers vGPU support, this functionality has had less time to be integrated into ESXi than the corresponding support in XenServer. The Nvidia vGPU support in KVM comes through RedHat Enterprise, but no free version is currently available, and this support is a recent addition to KVM.

3.3 Remote Desktop

Remote desktop is the type of software used to connect thin clients to the VMs to play games. Important factors to consider when selecting remote desktop software are clarity, responsiveness, flexibility, and compatibility.

Clarity is the visual fidelity of the game and is affected by the compression algorithm used for transmitting the video. The most important factor in the compression is the bitrate (the quantity of data transmitted per second). The bitrate affects the number of pixels that can be updated each second. If the image displayed is rapidly changing then at low bitrates the screen will appear pixelated and blurred, which is unacceptable when gaming.

Responsiveness is the latency between sending input and seeing the result on the screen. The lower the compression, the higher the latency tends to be as it takes longer to encode, send, and display each frame. Higher clarity always leads to higher response times and as such response times must be ranked against clarity.

Flexibility refers to the different games and applications that are supported when using the remote connection.

Compatibility is the different platforms the remote software can be used on, as well as the relative performance of different operating systems.

These factors were tested using different remote desktop software to connect from thin clients to the VMs running on the test servers with GPU resources. The VMs used for testing ran Windows 10 with League of Legends, Warframe, DoTA2, World of Warcraft, and Bioshock Infinite. The thin clients in the test lab ran Ubuntu 14.04, with a single thin client running Windows 10 to test an alternative OS. Table 3.4 summarises the performance of all remote desktop software tested based on the critical factors.

TABLE 3.4: Summary of remote desktop software performance

	Available Clarity Settings	Response	Flexibility	Compatibility (In brackets = poor performance)
XenDesktop	High/Low	Good	Limited	Windows, (Linux), Mac, Android, iOS
RemoteFX	Automatic	Excellent	Full	Windows, (Linux)
Steam Streaming	High/Low	Good	Steam Apps	Windows, Linux, Mac
GamingAnywhere	Full Control	Average	Full	(Windows), (Linux)
TeamViewer	Automatic	Average	Limited	Windows, Linux, Mac, Android, iOS
VNC	Automatic	Average	Limited	Windows, Linux, Mac, Android, iOS

Overall XenDesktop shows potential, with the recent software updates, to be a good paid option for a cloud-based internet cafe. However, the testing license for XenDesktop expired leaving RemoteFX as the clear winner out of the options that were available during testing due to its ability to run games, i.e., Excellent Response and Full Flexibility.

The remainder of this section describes the performance of each of the remote desktop software options in more detail.

3.3.1 Citrix XenDesktop

Citrix XenDesktop is a software package which both creates remote desktop connections and manages the VMs used with the remote desktops.

Citrix provided licenses to test XenDesktop and sent an engineer to help with installation. This was extremely useful as XenDesktop requires a full Microsoft ActiveDirectory set up to

function and uses Windows accounts to authenticate connections. ActiveDirectory utilises Windows Server, an OS that was not present in the test setup (note the test setup mostly uses Linux as the OS). Once a Windows Server VM was set up to run XenDesktop it was connected to the test servers, then base images for the gaming VMs were created. The VM images are identical except for the vGPUs that each image has. A base image is duplicated each time a new user connects to that image, and a unique VM is started up for that user. XenDesktop is set to run extra buffer VMs by default, so users do not have to wait for a VM to start up. This streamlining comes with the requirement that sufficient storage is available to handle the duplicate vDisks.

Once base VM images are set up, XenDesktop uses a simple web interface to allow users to find and connect to VMs using an ActiveDirectory account. These accounts can also be configured with permissions that determine the VMs that are available for connections. Connecting to a VM is simple and once connected the experience was high clarity with good response times. Unfortunately, settings for video quality are hidden away in the Windows registry.

Although XenDesktop smoothly starts up and runs games, the games proved impossible to control with a mouse or controller as it is not possible to give a VM full capture of a device. Instead, the device's inputs must pass through the local machine and then on to the VM. This extra "layer" causes problems in most games which use relative mouse movement rather than absolute mouse movement. Absolute mouse movement is the standard desktop movement and involves the mouse having a specific position on the screen, and all movement relates to an absolute position on the screen. By contrast, with relative mouse movement the mouse position changes per frame. This approach enables the cursor to be hidden, sensitivity to be adjusted and movement in a game to be smooth and clean. The game window must be able to fully capture all mouse inputs, to access relative mouse movement. Since XenDesktop passes mouse inputs via the local machine, relative mouse movement cannot be utilised. Unfortunately, this technical issue renders most games completely unplayable.

In addition, XenDesktop's Linux application (that was deployed on the thin clients) had performance issues with frequent connection errors and crashes.

3.3.2 RemoteFX

RemoteFX is Microsoft's standard remote desktop connection with support for DirectX and OpenGL. RemoteFX is automatically enabled on Windows versions newer than Windows 7. Remote desktop is a Microsoft product and, as such, isn't officially supported on Linux. Fortunately, there are many unofficial open source options for remote desktop connections. For our test setup, we used Remmina. Remmina's latest stable release worked on Ubuntu 14.04, supported RemoteFX and offered many quality settings. The default Remmina version that came with Ubuntu 14.04 did not support RemoteFX, and an updated version was required to test gaming.

Remmina and RemoteFX support full mouse capture by the destination [VM](#) which enabled games to use the mouse with relative mouse movement. Despite supporting remote desktop, it is clear that Remmina is not a supported application for using RemoteFX with latency and framerate problems apparent when playing games. This is unfortunate, as remote desktop and RemoteFX are extremely clean and stable when using a Windows machine for the thin client that connects to a [VM](#). Of course, it is not ideal from a commercial viewpoint to have to pay for a Windows license on both the thin clients and the [VMs](#).

3.3.3 Steam Streaming

Steam Streaming is a service offered on Valves Steam platform [138]. When the same Steam account is active on the same network on multiple machines, each machine gains the option to stream installed games from the other machine. To enable Steam Streaming in the test setup, the Linux Steam client was installed on the thin client, and the Windows Steam client was installed on the [VM](#), both with the same Steam account. Test games were then installed

via Steam on the [VM](#). Once games are installed it takes a single click to start playing the desired game with minimal problems starting the game.

Steam Streaming provides settings to balance quality and bandwidth. Unfortunately, even when set to high quality, this approach suffered from compression artefacts, and in fast-paced games, it was difficult to ascertain what was happening. In addition, while it is possible to add non-Steam games to Steam and stream them, this approach would often crash the game or cause incorrect window rendering making the games unplayable.

3.3.4 Other Remote Desktop Software

The remaining remote desktop options which were tested are discussed here. All of these options had serious issues that made them non-viable.

GamingAnywhere is an open source remote gaming platform made by Huang et al. [[67](#), [72](#)]. GamingAnywhere attempts to provide a highly extensible, portable, and, configurable cloud gaming platform. It includes support for Windows and Linux. Getting GamingAnywhere operational in our virtual environment was difficult, and once it was operational, its functionality was extremely unreliable, with the Linux functionality being mostly absent. As the last update to GamingAnywhere was in January 2015, it is in need of updates and reliability improvements to make it a functional platform for a cloud-based internet cafe.

TeamViewer is a support application that allows someone to take control of a computer remotely. TeamViewer has been suggested as an option for remote gaming on forums online, but in our test setup, it was not found to be fit-for-purpose with the Linux application being unreliable and gaming itself being unpleasant, due to low visual quality when frames are changing rapidly (i.e., a low bitrate).

Virtual Network Computing ([VNC](#)) is a graphical desktop sharing system that uses a remote frame buffer protocol to connect remotely to other computers. [VNC](#) is the standard remote connection for Linux systems and is comparable to Remote Desktop on Windows.

Unfortunately, it suffers from the reverse problem to RemoteFX, where the Windows VM end of the connection is unstable. However, the biggest downside of VNC is that it only supports OpenGL and not DirectX. This is because DirectX is a Windows driver. Unfortunately, DirectX is also the engine most games use, which makes VNC unsuitable for use in a cloud-based internet cafe.

3.4 Environment Testing

After investigating the available hardware (see §3.1) and software options (see §3.2, 3.3) for the prototype cloud-based internet cafe, the cloud-based internet cafe environment was tested. This testing was conducted to determine the performance of the environment with hardware as described in §3.1 and software as described throughout this section. The main purpose of this testing is to investigate the feasibility of the test setup environment as a cloud-based internet cafe environment.

For these tests, the servers were running XenServer 5.1 with Citrix XenDesktop 5.6 as the VDI. The performance benchmarking was conducted using 3DMark. 3DMark [139] is game benchmark software that tests a computer's performance under standard game conditions with a variety of levels of rendering, lighting, and physics effects.

The 3DMark software provides four tests: IceStorm, a basic benchmark; CloudGate, a medium DirectX 9 benchmark; SkyDiver, a medium DirectX 10 benchmark; and FireStrike, a difficult DirectX 10 benchmark. A score is given based on the framerate throughout the tests. Each test includes two render/lighting tests and one physics test, with SkyDiver and FireStrike including an additional combined test. For the cloud-based internet cafe, we categorise games into low, medium, and high requirement games. The four 3DMark tests map to these levels as follows: IceStorm: low; CloudGate: medium; SkyDiver: medium-high; FireStrike: high.

The tests were conducted first for a single VM running on the server with a variety of different VM specifications. Next, testing was carried out with multiple VMs with the same specifications running on a single server to provide a measurement of how performance changes as VMs are added to a GPU core.

3.4.1 Single Machine Testing

Extensive testing was conducted with a single VM running on a single server. The VM was given differing quantities of CPU, RAM, and different vGPUs. Test results are used to quantify each server's performance.

Using XenDesktop as the virtual machine management and connection program, a virtual machine template was created running Windows 7 with Nvidia drivers and 3DMark installed. This template was then used to create a set of VMs with each vGPU type, 1, 2, 4, 6, or 8 CPU cores and 2, 4, 8, 12, or 16GB of RAM. The four 3DMark tests were then run on each of these machines using the default settings, and the 3DMark scores were recorded. Each test was run independently of the others, so they did not influence the performance of one another. After each test, the VM was changed so that all combinations of vGPU, CPU and RAM configurations were tested. This testing resulted in a matrix of results showing the difference arising from changing the vGPU (consisting of part of a GPU core and vRAM), a number of CPU cores, and amount of RAM. Each configuration was tested once, taking around 30 minutes to complete all four 3DMark tests.

The test results for each 3DMark test were fitted with linear regression models, with all models yielding adjusted R squared values over 80%, meaning that all fitted models do an excellent job of explaining variation observed in the data. The covariates for all these models were the number of CPU cores and the vGPU type. In all the tests we considered including RAM as an additional covariate, but the corresponding fitted coefficient was not significantly different from 0, indicating that RAM was not needed in the model to explain the test score, i.e., the test score did not differ significantly over the amounts of RAM tested. As RAM

does not affect the test scores, the presented test results were averaged over all the different **RAM** amounts to produce a single average score for each **vGPU** and number of **CPU** cores combination.

The results show that the Nvidia **vGPU** drivers put a cap on the framerate of 67FPS and hence limit the maximum achievable score. This cap does not apply to the pass-through **GPU** (no virtualisation) which has an uncapped framerate.

The results also show a significantly higher score for all tests with the K2 server over the K1 server, as expected, due to the different Kepler cores. The only real difference between the **vGPUs** of each core type is the amount of available **VRAM** because they were the only **vGPU** on the **GPU** core during testing. Results show that for Ice Storm, Cloud Gate, and Sky Diver the **VRAM** makes no difference to the average score. However, the uncapped framerate on the pass-through **GPUs** allows them to fully utilise the core's capability with the K1 server achieving higher scores for Ice Storm, and the K2 server achieving higher scores for Ice Storm and Cloud Gate with no score difference for Sky Diver compared to all **vGPUs**. The Fire Strike test shows the effect of **VRAM** on performance: the K100 and K200 cards are unable to run the test due to insufficient **VRAM**. The K120 shows a lower score than the other **vGPUs** available on the K1 server. There was no evidence of a difference between the other **vGPUs** available on the K1 server. The K220 also showed a lower score than the other **vGPUs** available on the K2 server. Similar to the K1 server, there was no evidence of a difference between the other **vGPUs** available on the K2 server.

The model showed that **RAM** makes no significant impact on the score for any **vGPU** or **CPU** independent of the test (p-value > 0.1). Since graphics rendering mainly uses **VRAM** for loading art, it is not surprising that **RAM** would have no impact on the achievable score. **CPU** is a significant factor towards scores achieved: as the number of **CPU** cores is increased the scores in all tests increase. At one **CPU** core, the virtual machine is barely functional, and scores degrade significantly. Between 2 and 8 **CPU** cores, there is a roughly linear decrease for each **CPU** removed.

Figure 3.7 shows average scores for each **vGPU** type and **CPU**. **RAM** was not included in the figure as it was not a significant factor. Each parallel bar represents a test score, from left to right: Ice Storm, Cloud Gate, Sky Diver, and Fire Strike. The graphs show far higher scores for the pass-through GRID K2 and a high Ice Storm score for pass-through GRID K1. The GRID K2 vGPUs outperform the GRID K1 vGPUs in all but Ice Storm where both hit the 67FPS score cap.

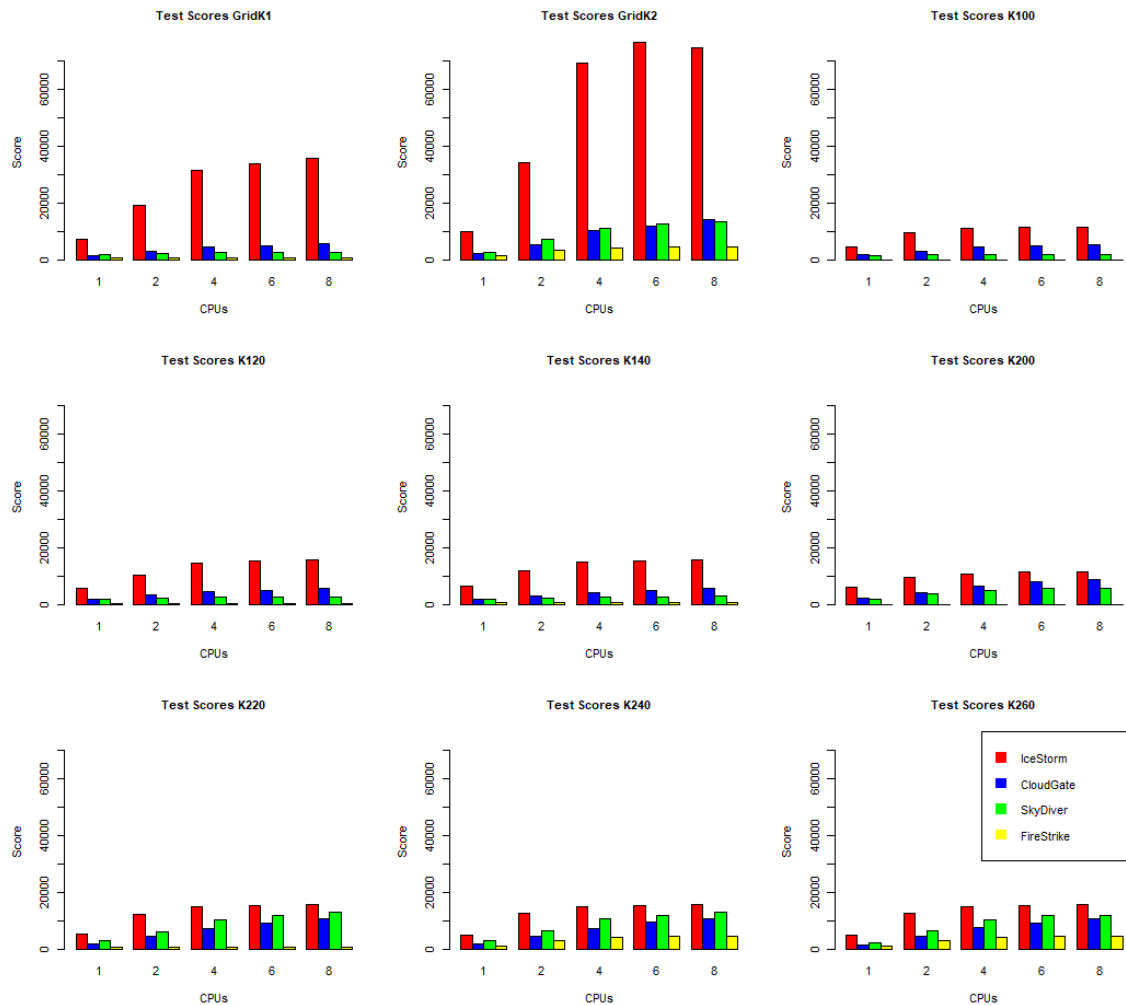


FIGURE 3.7: Grid of 3DMark scores for **vGPU** and **CPU** configurations. Tests from left to right bars: Ice Storm; Cloud Gate; Sky Diver; and Fire Strike.

3.4.2 Parallel Testing

In practice, multiple machines will be running simultaneously. Quantifying the effect of running multiple machines further informs us of the servers' performance.

It is important to test the effect of running multiple copies of the 3DMark tests on a single physical GPU core. If the core is overloaded, then processes must be queued (as described in §3.1.2.1) and performance will deteriorate. To test this, multiple vGPUs were allocated to the same core, and 3DMark was run simultaneously on each machine. Results for each VM on the core were recorded and compared to equivalent results for a single machine. Two VMs were tested in parallel for all configurations of 4 and 8 CPU cores; 4 and 8GB of RAM; and K140, K240 and K260 vGPUs. Four VMs were tested in parallel with: 4 and 8 CPU cores; 4 and 8GB of RAM; and K140 vGPUs. Fewer configurations were required to be tested than for single machines as the performance changes are related to the total load on the GPU core and as such can be predicted for other configurations.

Parallel testing produced interesting results. Due to the framerate cap in the Nvidia drivers, the GPU cores were not fully loaded in some tests. This spare capacity allowed parallel processing with lower performance loss than would be experienced with uncapped framerates. Results show a maximum achievable score that is equal to that for a single, same specification, machine, but this maximum drops as the GPU core become overloaded due to either more virtual machines on the GPU or a higher performance test. Results showed that the K1 server could provide multiple VMs playing low-end games and the K2 server could provide multiple VMs playing low and medium end games.

As Table 3.5 shows, the K140 vGPU showed a statistically insignificant change (i.e. within single machine testing score variation) in Ice Storm score when a second VM was tested in parallel and only an approximately 20% score drop with four VMs. In the other tests, however, adding a second VM caused around a 50% drop in score. A further 30% drop in score occurred for all tests when four VMs were run in parallel. Interestingly, this drop is

TABLE 3.5: Test results for the K140 vGPU (Percentages show difference from single machine)

CPU Cores	No. Machines	IceStorm	CloudGate	SkyDiver	FireStrike
8	1	15,890	5,634	2,859	797
8	2	15,672 (-1%)	2,943 (-48%)	1,482 (-48%)	576 (-28%)
8	4	12,213 (-23%)	1,603 (-72%)	836 (-71%)	241 (-70%)
4	1	14,858	4,385	2,711	781
4	2	14,725 (-1%)	2,748 (-37%)	1,374 (-49%)	550 (-30%)
4	4	12,062 (-19%)	1,486 (-66%)	882 (-67%)	286 (-63%)

less significant despite doubling the number of processes on an already overloaded GPU core.

This may be the result of the intelligent allocation of processes to the GPU core.

TABLE 3.6: Test results for the K240 vGPU (Percentages show difference from single machine)

CPU Cores	No. Machines	IceStorm	CloudGate	SkyDiver	FireStrike
8	1	15,797	10,840	13,098	4,468
8	2	15,643 (-1%)	10,367 (-4%)	8,640 (-34%)	2,523 (-44%)
4	1	15,018	7,282	10,614	4,253
4	2	15,093 (0%)	7,536 (3%)	7,408 (-30%)	2,491 (-41%)

Table 3.6 and 3.7 show that the K240 and K260 vGPUs showed a statistically insignificant change in score from the single machine for the Ice Storm, or Cloud Gate tests when a second VM was added. Sky Diver suffered a 30% decrease in score and Fire Strike suffered a 40% decrease in score with the second VM.

TABLE 3.7: Test results for the K260 vGPU (Percentages show difference from single machine)

CPU Cores	No. Machines	IceStorm	CloudGate	SkyDiver	FireStrike
8	1	15,794	10,735	11,866	4,551
8	2	16,033 (2%)	10,550 (-2%)	8,417 (-29%)	2,461 (-46%)
4	1	15,009	7,911	10,337	4,551
4	2	15,436 (3%)	8,230 (4%)	7,693 (-26%)	2,573 (-43%)

3.4.3 Summary

This section summarises the results and places them in the context of real-world performance which enables the construction of test data sets described in §4.3.

Placing the 3DMark scores in real world context requires the comparison of the K1 and K2 servers with a real-world high-performance PC. The comparison is best set at a baseline level which is the maximum performance of the cards. In order to generate the maximum performance potential of the K1 and K2 cards, both the K1 and K2 cards were setup with only a single VM running on one of their cores. Running a single VM on a core shows the maximum potential performance of the cards. Table 3.8 shows 3DMark scores of these VMs compared to a high performance desktop PC with a Nvidia GeForce GTX 970, and an I7-6700 CPU capable of running Witcher 3 on high settings at 1080p with over 60FPS. The 3DMark scores show the K1 performing acceptably on IceStorm, with increasingly poor performances on CloudGate, SkyDiver, and FireStrike. The K2 can achieve excellent scores on IceStorm, CloudGate, and SkyDiver, with a significant fall-off in performance on FireStrike but still maintaining an acceptable score. However, the high-performance machine can almost double the score of the K2 card even at its highest possible performance.

The linear regression models that were fitted to analyse the significance of each VM component found that: 1) RAM had no impact on performance; 2) VRAM had minimal impact; 3) the number of CPU cores and the GPU core speed were the most critical factors affecting 3DMark scores and hence video game performance. In particular, it was found that the K2 server performed better than the K1 server, and performance improved as the number of CPU cores increased.

Combining the linear regression model with the baseline performance comparison finds that overall, neither the K1 or K2 card meet the performance requirements of a high-end internet cafe PC in 2017 with the K2 card showing the potential to provide a medium or low-end gaming VMs. A newer generation of cards is required to meet the modern performance requirements of an internet cafe. However, in 2015 both the cards had the ability to play a

TABLE 3.8: 3DMark scores of K1 and K2 vs Gaming PC

Cores	CPU Clock	RAM	GPU	IceStorm	CloudGate	SkyDiver	FireStrike
8	3.4GHz	8GB	GTX 970	141,057	26,504	24,991	9,484
8	2.6GHz	8GB	K2	74,562	14,318	13,379	4,550
4	2.6GHz	8GB	K2	75,688	11,589	11,736	4,291
2	2.6GHz	8GB	K2	38,836	5,664	7,624	3,403
8	2.6GHz	8GB	K1	33,461	5,480	2,778	794
4	2.6GHz	8GB	K1	31,792	4,520	2,656	782
2	2.6GHz	8GB	K1	19,449	3,015	2,404	749

variety of games, and this base level is assumed when building test sets as this is when the technology was relevant. The values for the benchmarks assuming 2015 gaming demand are used to build test data sets described in §4.3 for the resource allocation algorithms.

This chapter described the hardware and software technology needed to construct a cloud-based internet cafe and benchmarked this technology. This benchmark allows specifications to be generated for VMs which meet the demands of internet cafe users. These VMs are then used as inputs when testing algorithm performance as part of the larger test data sets. These test data sets also utilise the server setups described in this chapter. In particular all servers are assumed to run Citrix XenServer, with RemoteFX connecting a Windows VM with a Windows desktop.

Chapter 4

Allocation Problem

This chapter defines the resource allocation problem for the cloud-based internet cafe considered in this thesis and describes the inputs, decisions, outputs, constraints, and assumptions. This chapter also describes the data sets built for testing the algorithms defined in chapters 5, 6, and 7.

4.1 Problem Description

The resource allocation problem for a cloud-based internet cafe considers the acceptance of users and the placement of those users onto a given set of servers. The objective for this problem is to maximise the total profit gained from all users.

Users or customers arrive in the internet cafe throughout the day and then request to stay for a number of hours. These users arrive at discrete time points (in the test sets in this thesis either every hour or 15 minutes). They stay for a number of time points before leaving (in this thesis they stay for a whole number of hours). While a user is staying in the internet cafe, they demand one of the services offered in this problem: either web, low, medium or high. Each service has a different price per hour with the price increasing as the resources required increases. An accepted user will consume a proportion of a server's resources to

access their service via a VM and take one of the available seats in the internet cafe. Once a user has been placed on a server they are fixed to that location and cannot be moved. For this reason, when servers are near capacity or most of the internet cafes seats are taken it may be beneficial to turn away a user demanding a less profitable service or who is staying for a shorter duration in anticipation of a more profitable user. Also, the decision to accept or reject a user must be made immediately and cannot be delayed. If a user is rejected, they give no profit.

The four services represent a flexible breakdown of the different applications offered in an internet cafe. The web service uses the least resources and is for users demanding web browsing, email, or video. The low, medium and high services are for users demanding games of increasing resource requirements. The specific games which fall into each category are described for the specific test sets in §4.3.

Once a user is accepted, a VM is created for them which is capable of providing the demanded service. These VMs are created on a specific server and GPU core. The users are partitioned into two sets: gaming, and web. Each type of VM uses a fixed proportion of a server's CPU and RAM resources, using a number of cores, and gigabytes (GBs) of RAM respectively. Web VMs do not use any GPU resource and share their cores and RAM between web users. This sharing means a single web VM can supply multiple web users. Gaming VMs use GPU resource and have exclusive use of their CPU cores and RAM. The VM serves either low, medium or high-end gaming users. The GPU resource consists of a portion of a GPU core's processing power and VRAM, which is determined by the vGPU type as described in §3.1.2.1.

The CPU cores and RAM are virtualised such that 1 virtual CPU core has one corresponding physical core and 1 virtual RAM chip has a corresponding physical RAM chip. In the case of web machines they are virtualised in the same 1:1 ratio but are overallocated between the web users on the assumption no one user will be using the full quantity of CPU and RAM allocated to the VM.

The GPU's are virtualised according to the rules described in §3.1.2.1. These rules mean each user must be placed on a server and GPU core and once placed the VM will lock the core to a given vGPU type until all VMs on that core are removed. To model this, GPU cores are considered to be virtualised into discrete configurations. Each configuration runs a number of VMs supplying various services. These VMs must all have the same vGPU type, which then defines the vGPU type of the configuration. Each configuration uses a quantity of CPU and RAM equal to the total required by the VMs in that configuration. At any point in time, a GPU core may be assigned at most one configuration. If no VMs in the configuration are in use at that point in time then the core may switch to any other compatible configuration. However, if one user or more is currently using a VM running on the core, then the core may only be swapped to a configuration with the same vGPU type as its current configuration. In addition, the new configuration must include VMs which can supply the services demanded by all users currently active on the core.

TABLE 4.1: Example VMs for a K2 server

Service	CPU	RAM	GPU	Minimum vGPU
Low	4	6	0.25	K240
Med	6	8	0.5	K260
High	8	10	1	K280

TABLE 4.2: Example of configurations for a K2 server

Configuration	CPU	RAM	GPU	vGPU	Low	Med	High
K240 Low 1	4	6	0.25	K240	1	0	0
K240 Low 2	8	12	0.5	K240	2	0	0
K240 Low 3	12	18	0.75	K240	3	0	0
K240 Low 4	16	24	1	K240	4	0	0
K260 Med 1	6	8	0.5	K260	0	1	0
K260 Med 2	12	16	1	K260	0	2	0
K260 Low Med	10	14	0.75	K260	1	1	0
K260 Low 1	4	6	0.25	K260	1	0	0
K260 Low 2	8	12	0.5	K260	2	0	0
K280 High	8	10	1	K280	0	0	1

Table 4.2 shows example configurations as used in the model. In this example low users are assigned a VM with 4 CPU cores, 6GB of RAM, and one quarter of a GPU core, on a K240 vGPU. Medium users are assigned a VM with 6 CPU cores, 8 GB of RAM, and half

a GPU core, on a K260 vGPU. High users are assigned a VM with 8 CPU cores, 10 GB of RAM, and a whole GPU core, on a K280 vGPU. These VMs are summarised in table 4.1. The vGPU types used are for a Nvidia GRID K2 GPU card as defined in table 3.2. Configurations are set up for all combinations of VMs with total GPU usage of up to 1 core. Configurations which do not use an entire core are considered as they must share the CPU and RAM with configurations running on other GPU cores. If the server only has 20 CPU cores available then always running to maximise GPU usage would result in not being able to run VMs due to limited CPU cores. Configurations are also set up for running low VMs on K260 vGPUs as these provide sufficient speed to meet the low VMs' requirements and allow the addition of medium VMs. This allows configurations with a mixture of low and medium VMs on the GPU core simultaneously, albeit with a lower maximum number of low VMs. However, medium VMs can not be run on K240 vGPUs as this would not meet their minimum requirements.

The limitations of GPU virtualisation are worth noting as playing high-end video games often needs large quantities of VRAM. When testing the servers, it was found that the GPU cores are not powerful enough to run large numbers of gaming VMs per core. The low power of the GPU cores plus the fact that the cores have double the VRAM of a standard desktop GPU make the GPU core speed the bottleneck of the GPU rather than VRAM. This bottleneck means it is possible to run the core with a lower vGPU type while still providing the VRAM needed to play the games, providing sufficient core speed remains available. Despite the fact that the restrictions on switching vGPU types are not significant in the real environment they still exist practically. As such a model for allocating resources to customers which handles these restrictions is presented in §5.1. A faster model which assumes the restriction is not a limiting factor and produces the same solutions when that is true is presented in §5.3.

4.2 Models

Allocating server resources for a cloud-based internet cafe requires a model capable of making intelligent choices about which customers to accept and how to place them on servers.

Any model must have inputs of 1) an arrival process for users in an internet cafe, 2) a departure process for users in an internet cafe, 3) the services or requirements which will be demanded by those users, 4) the servers that will run the VMs for the services and 5) the specifications of the virtual machines needed to run those services. The model must then output user allocations to servers, subject to the restrictions described in §4.1.

4.2.1 Notation conventions

The models use the following convention to differentiate between parameters and variables. Parameter p determined by index/expression i and j is denoted $p(i, j)$. Variable v indexed by indices/expressions i and j is denoted v_{ij} .

The notation for this problem is overloaded such that the symbol for an element in a set (e.g., $x \in X$) is also used to refer to objects in that set. For example x alone refers to an item in set X but may also be used as $x(i)$, the element of x which is associated with $i \in I$. The symbol may be used for such references multiple times, E.G., $x(j, k)$, the element of x associated with $j \in J, k \in K$. This overloading is done to improve the readability of the equations.

All notation conventions discussed here are summarised in the notation section at the beginning of the thesis.

4.2.2 Inputs

The inputs for this model consists of (the arrival and departure of) users, services demanded by users, servers to supply VMs, VMs to supply the services, and configurations to define which GPU cores run which VMs.

4.2.2.1 Services

Services are demanded by users. In this problem, services are grouped into web, low, medium, and high. The web service provides all non-gaming activities. Low, medium and high provide gaming with each level containing increasingly resource-intensive games. Users pay an amount per hour to use the service and expect it to meet a minimum performance requirement. The services are provided by VMs on servers which have enough computing resources to meet or exceed the minimum performance requirement.

This leads to the definition of R , the services the internet cafe will be supplying. This set defines the services users will demand, their requirements, and profits. Each service $r \in R = \{\text{web, low, med, high}\}$ has:

- a profit $\pi(r)$ per hour;
- a list of servers $z(r)$ which can provide a VM meeting the minimum performance requirements for the service, $z(r)$ a subset of all available servers S .

4.2.2.2 Servers

Servers supply VMs for the demanded services. Each server has a type of GPU core and a number of: CPU cores, GBs of RAM, and GPU cores. These server resources are virtualised and divided amongst VMs created for services demanded. Each server has a VM for each service, which uses a number of these CPU cores, GBs of RAM, and a proportion of a GPU core.

Feasible sets of VMs can be determined for a server, depending on its specifications. These sets of VMs are referred to as configurations. A server can run one configuration per GPU core. A configuration runs a number of VMs for each service. Each of these VMs has the vGPU which is defined by the configuration. In addition the total GPU used by these VMs

cannot exceed 1. Overall these configurations define the VMs running a server at any point in time. See table 4.1 for examples of VMs and table 4.2 for examples of configurations.

Web services do not require any GPU resources at all and have unique non-core configurations run directly on the server. In addition, a web user uses little CPU and RAM resource with some exceptions. These exceptions arise from high spikes in resource usage when opening a new web page or initial buffering of a video. These spikes mean web users require much higher resources in some moments than during the majority of their stay. For this reason, it makes sense for web users to share resources with each other. This is done by grouping web users on a special shared VM which handles multiple users simultaneously.

This results in a list of servers S which will supply users with the requested service. Each server $s \in S$ has:

- resource capacities $c(s, q)$ where $q \in Q \equiv \{\text{cpu}, \text{ram}, \text{gpu}\}$ (the set of resources);
- a type of GPU core from the list of available GPU core types $g(s) \in G$
- a numbered list of cores $K(s) = \{0\} \cup \{1, \dots, c(s, \text{gpu})\}$, where the 0 core represents a special “null” core for VMs where no GPU core is required.

The GPU cores being utilised by the servers are defined as G (as noted previously). Each of those GPU cores $g \in G = \{K1, K2\}$ has:

- A list $Z(g)$ of different vGPUs available on that core.

A list of vGPU types for Nvidia GRID K1 and K2 cards can be found in table 3.2.

4.2.2.3 VMs

The services R are provided by VMs running on the set of servers S . For each service $r \in R$ and server $s \in S$ there is a unique VM that provides the resources and performance required to

provide service r on server s , so VMs are denoted by pairs (r, s) . Each VM (r, s) $r \in R, s \in S$, defines:

- a type of GPU core $g(s) \in G$ needed to provide service r ,
- a type of vGPU core $\zeta(s, r) \in Z(g(s))$
- a quantity of each resource required $b(s, r, q)$, for each $q \in Q = \{\text{cpu}, \text{ram}, \text{gpu}\}$

In addition, some VMs may be shared between multiple users if the peak requirements are vastly different from the averages. In this problem there is a single group of this type (web users) that share resources. Hence, each VM also defines:

- a number of users that can share this VM $\bar{n}(s, r)$

A service may not require any GPU and as such some VMs may have $b(s, r, \text{gpu}) = 0$ and $\zeta(s, r) = \text{none}$.

4.2.3 Configurations

The VMs are combined into sets of configurations. Each configuration uses a single GPU core and runs a specific number of VMs of each type. Configurations are created for every combination of VMs with total GPU resource requirement less than or equal to 1.

The set of all feasible configurations is M . Each configuration $m \in M$ has:

- $v(m, r)$ a number of VMs supplied for each requirement $r \in R$;
- $\rho(m, q)$, for each $q \in Q$ a quantity of each resource used;
- $\zeta(m)$ the type of vGPU core used;
- $g(m)$ the type of GPU on which the configuration can be used, i.e., that contains cores of type $\zeta(m)$.

4.2.3.1 Users

The internet cafe users are the focus of all decisions being made. Each user arrives and departs at a certain time, and demands a service. The user pays for the service per hour at the rate set for the service.

Supplying the user's demanded service (referred to simply as the user's service, for brevity) requires a VM to be placed on a server. A quantity of resources is consumed on that server for the period the user is present. The resources consumed are constant over time for all test sets considered in this thesis. However, it is worth noting that the online algorithms (described in Chapter 7) can handle variation in the resources consumed over time.

It is necessary to define the points in time at which decisions regarding users might be made. The epochs $t \in T$ defines all the times at which a decision might be made. T is a discrete set of evenly spaced time points.

To model demand for services at an internet cafe, users that visit the cafe throughout the day are defined by the set U . Each user $u \in U$ has:

- an arrival time $a(u) \in T$;
- a departure time $d(u) \in T$ where $d(u) > a(u)$;
- a stay duration $j(u) = d(u) - a(u) > 0$;
- demand for a service $r(u) \in R$;
- a profit resulting from providing the demanded service to a user $p(u) = \pi(r(u)) \cdot j(u)$;
- a list of servers, $z(u) = z(r(u)) \subseteq S$ to which the user can be assigned.
- $b(s, q, u, t)$ the quantity of each resource required by user u at time t when placed on server s which is equal to $b(s, r, q)$ if $a(u) \leq t < d(u)$, 0 otherwise (the number of non-zero elements in this list is equal to $j(u)$). Note that $b(s, q, u, t)$ allows users demanding the same service to utilise different quantities of resources on the same server. However,

in all problems considered here the quantity of resources utilised is constant for any user of the same service on the same server.

Another useful (calculated) input is the maximum duration any user stays $J = \max_{u \in U} j(u)$.

Let E denote the set of unique times at which users arrive, ordered temporally. Note that E is a subset of T . Given an element $e \in E$, e^+ denotes the next element in E and e^- denotes the previous element in E . If $a(u) \leq e < d(u)$ then user u is present at event point $e \in E$. These event points only include user arrival times as these are the points when decisions are made, no choices are made when users depart or when no users arrive or depart therefore it is only necessary to consider the state of the system when users arrive.

Additionally, an internet cafe has a maximum number of users it can physically seat, \bar{N} .

4.2.4 Decisions

For each user $u \in U$ a decision must first be made whether to accept or reject the user. If the user is accepted then a server $s \in S$ and a GPU core $k \in K(s)$ must be selected to host the user's VM.

4.2.5 Constraints

Decisions that can be made are limited by a number of constraints on the servers and GPU cores.

First, at any point in time $t \in T$, no server $s \in S$ may have the total resources in use exceed the available resources $c(s, q)$ for all $q \in Q$. Additionally the total users in the internet cafe at any point in time $t \in T$ must not exceed the total seats in the internet cafe \bar{N} .

If a GPU core is running a VM with vGPU type $\zeta(s, r)$ then it is not possible to start VMs with a different vGPU type on that core. Note that since the server will be the same, it is the service that the VMs provide that differs and causes the difference in vGPU type. Hence,

any $\hat{r} \in R, \hat{r} \neq r$ with $\zeta(s, \hat{r}) \neq \zeta(s, r)$ cannot be started on the same GPU core. In addition, through consecutive points in time, the VM allocated to a user can not change.

4.2.6 Outputs

Our final outputs for all models are: an allocated server $s(u)$ for each user $u \in U$, and for each server $s \in S$ at each event point $e \in E$ for each service $r \in R$: a number of virtual machines running $\nu(s, e, r)$. This describes which users are accepted, on which servers they are placed, how server resources are allocated, and which virtual machines must be operating.

4.3 Test Data Set

Test data sets were generated for the problem described in §4.1 to test algorithms. These test sets are fully described for all inputs in this section. The problem test sets are divided into two distinct groups. The first was created a few years ago when games played had lower requirements, and as such, each VM requires fewer resources for each service level. The second set has higher requirements for each service matching current demands. Both test sets use the same servers as described in §3.1.1.

These tests always have a predefined total number of users for the entire test set. The fixed number of users provides a constant problem size so the changes in other factors can be more easily compared.

All data sets have four services offered: web, low, medium, and high. *Web* offers web browsing, emails, video watching. *Low* offers the lowest requirement games. *Medium* offers the higher requirement games which are either slightly older or new multiplayer games designed with performance in mind. *High* offers the latest games which are often single player. However, the price and exact games played on each service vary between the two sets.

4.3.1 First Test Set

The first test data set was made specifically for the offline integer program in order to test the potential gains in total resource utilisation between a traditional internet cafe and a cloud-based internet cafe. This data set considers a 10 hour period when the internet cafe is busiest between 3 pm and 1 am. Users in this test only arrive once an hour at the start of the hour.

Testing scalability of algorithms is important. Hence, three different sized internet cafes are tested; 30 seats, 150 seats, and 300 seats. The average internet cafe in China (the largest internet cafe market in the world) has 120 seats so this range covers the relevant internet cafe sizes. This test set is also used to test variability in services consumed in the internet cafe and is grouped into 6 test cases with three levels of web usage, and two levels of gaming split between low, medium and high. These demand levels are described in §4.3.4.

4.3.2 Services offered

The services offered in this test set are: web; low; medium; and high. These levels are as described in §4.3. For this test set, each level is defined by a minimum 3DMark score obtained in §3.4. Web machines obtained a score of at least 5,000 in IceStorm, low machines obtained a minimum score of 10,000 in IceStorm, medium machines reached a score of at least 1,400 in CloudGate, and high machines had a minimum score of 2,000 in FireStrike. The 3DMark scores must not fall below these minimum values even when the server is fully loaded with VMs.

The services are priced according to the resources used, with each service level requiring twice the GPU resource of the previous level and, as such, twice the price. These prices and 3DMark scores are shown in table 4.3.

TABLE 4.3: Profit and minimum 3DMark scores for test set one services

Service	Price Per Hour	3DMark Test	Minimum Score
Web	\$ 1.00	IceStorm	5,000
Low	\$ 2.00	IceStorm	10,000
Medium	\$ 4.00	CloudGate	1,400
High	\$ 8.00	FireStrike	2,000

4.3.3 Server Inputs

In this test set, there are two types of servers that supply VMs for the services. These are the servers described in §3.1.1. These servers have either a K1 GPU or a K2 GPU card, called the K1 server or K2 server respectively. Each server has 32 CPU cores and 80GB of RAM. The number of servers used to provide services to users is described in table 4.4 where the quantity is determined by the total number of seats/users and the gaming demand level. These server quantities are set such that the internet cafe can supply all seats in the internet cafe with VMs if demand is average.

TABLE 4.4: Test set one Internet Cafe Sizes and Server Quantities

Size	Gaming Demand Level	Total Users	Seats	Number of K1 Servers	Number of K2 Servers
Small	Low	100	30	1	1
	High	100	30	1	1
Medium	Low	500	150	5	3
	High	500	150	3	5
Large	Low	1000	300	10	5
	High	1000	300	5	10

Each server runs VMs with specifications that give the minimum 3DMark scores even when the GPU core is full of VMs running in parallel. Table 4.5 shows these VM specifications for the K1 and K2 servers. Each web VM supplies 8 users with each user using on average a single CPU core. As a single CPU core is not able to give the minimum 5,000 IceStorm score, the web users instead share 8 cores amongst them allowing any one VM to reach the minimum score at any point in time, on the assumption that web users do not need this speed often.

TABLE 4.5: Test set one virtual machine specifications

		K1 Server				K2 Server			
Service	Users Per VM	CPU	RAM	vGPU	GPU %	CPU	RAM	vGPU	GPU %
Web	8	8	16	none	0	8	16	none	0
Low	1	4	6	K140	0.25	4	6	K220	0.125
Medium	1	6	8	K140	0.5	6	8	K220	0.25
High	1	N/A	N/A	N/A	N/A	8	8	K240	0.5

Configurations are the number of VMs of each service run on each GPU core where the total GPU % must be less than or equal to 1. In addition, all VMs in a single configuration must run the same vGPU type. A VM can be run on a vGPU type with a higher performance value than is given in its specifications but not lower. This may allocate a higher GPU % to that VM than the service requires. Example configurations include (on the K1 server): 4 Low VMs; 2 Medium VMs; or 2 Low VMs and 1 Medium VM alongside 5 more configurations. The K2 server can have configurations with; 8 Low VMs; 4 Medium VMs; 2 High VMs; 2 Medium VMs and 1 High VM; 4 Low VMs and 2 Medium VMs; 4 Low VMs and 1 High VM or any of the other 38 combinations. These combinations must obey the rules of vGPU usage as stated in §3.1.2.1. A full list of configurations can be found in appendix A.

Web VMs, which don't use vGPU resources, have configurations for the utilisation of all 32 CPU cores on the servers. Hence web configurations have up to 4 web VMs supplying up to 32 users.

4.3.4 User Inputs

Users for this test set require an arrival time, a number of hours to stay, and a service to demand.

No hard data was available on the exact usage patterns of internet cafe users. Fortunately, plenty of surveys exist that discuss the habits of users as discussed in §2.1. To generate data two surveys were used, consisting of 284 users in Malaysia[4], and 336 users in Turkey[25]. These surveys have a large enough sample size to be representative of usage patterns in their

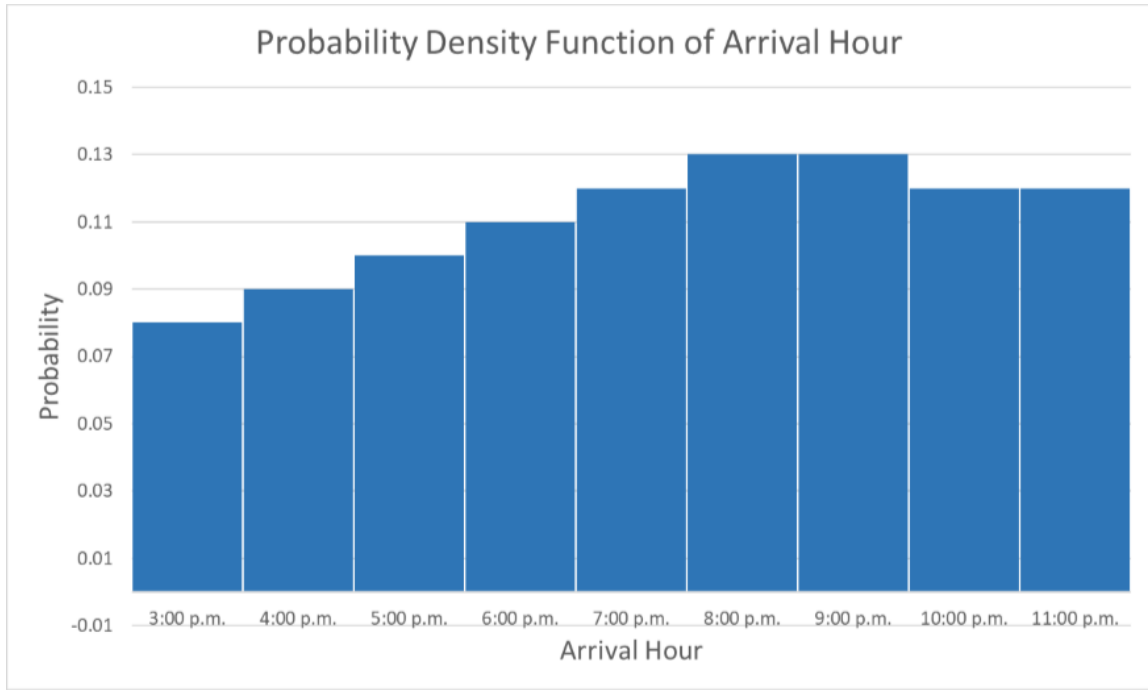


FIGURE 4.1: Probability density function for hours users arrive in test set one

geographic regions. These surveys divided user arrival times into five times of day: morning, midday, afternoon, evening, and late night. Few users used the internet cafe in the morning, with the quantity increasing throughout the day, peaking in the evening and falling off later at night. As we are interested in the higher demand times for resource efficiency analysis, the time periods considered are from 3 pm until 1 am or from mid-afternoon until early late night. The chance of any particular user arriving at any hour is shown in figure 4.1.

The surveys found that the average user stays between 2 and 3 hours, with some staying for longer duration's. A distribution of hours stayed was created using a gamma distribution and is shown in figure 4.2.

These surveys showed between 40%, and 60% of users would demand the web service, with the remaining playing games. Three groups were made, with either 30%, 50%, or 70% of users demanding the web service.

Neither survey specified the games being played. To generate gaming service demand for either low, medium or high gaming services, reports from AMD's Raptr software from 2015 were used [140]. AMD Raptr is driver management software for AMD GPUs. The games in

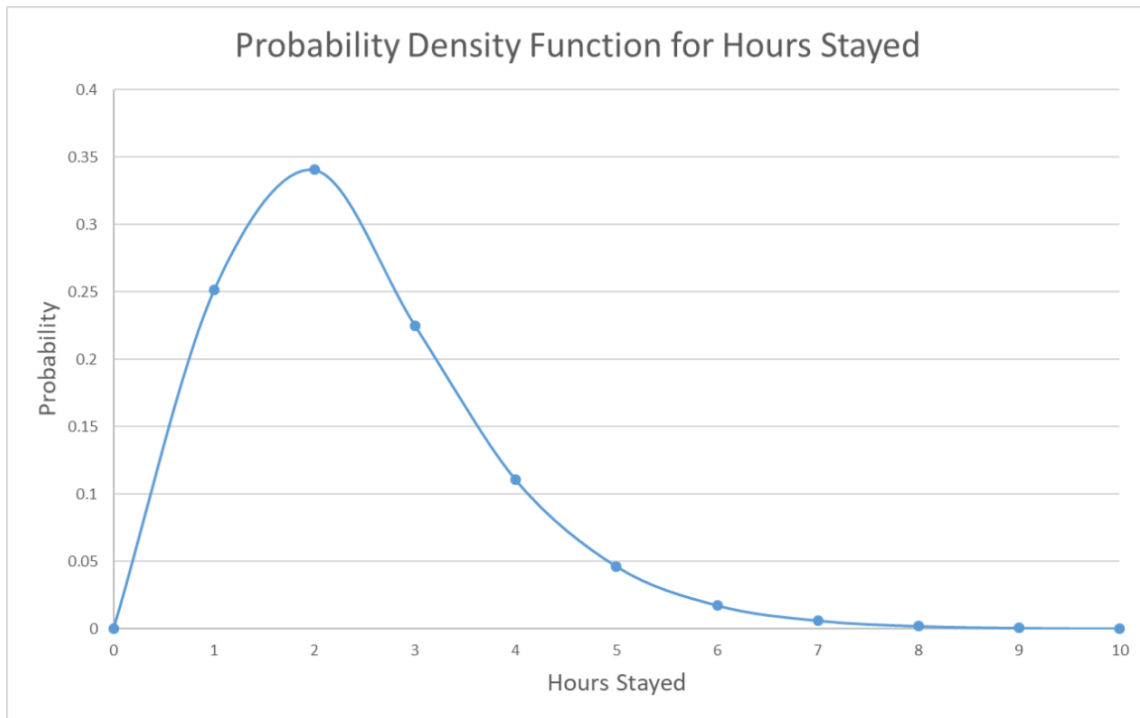


FIGURE 4.2: Probability density function for duration of stay in test set one

this list were divided into low, medium, and high based on their recommended hardware. The 3DMark scores can be linked to performance capabilities of different generations of Nvidia GPU cards. This link was used to set the score thresholds based on table 4.3. The low (gaming) service was set as requiring a GPU from Nvidia's 300 series cards or lower. The medium service was set as requiring a GPU between Nvidia's 700 series and 400 series. The high service was set as requiring a GPU from either Nvidia's 900 series or 800 series. This categorisation leads to the top 20 most played games split with: 34.5% at low, 21.12% at medium, and 6.31% at high. The remaining 38% of gaming hours were not spent on these top 20 games. Demand not in the top 20 was used to create two groups: one where demand kept the top 20 proportions, and one where the demand for medium and high increases proportionally. The first group gave demands of 56% for low, 34% for medium, and 10% for low giving mostly low gaming demand. The second group gave demands of 41% for low, 34% for medium, and 25% for high giving greater high gaming demand.

Combining these differing web demands and gaming demands creates 6 test scenarios. These scenarios are summarised in table 4.6.

TABLE 4.6: Proportion of demand for each service for test set one

	Percentage Demanding Service in Scenario					
	S 1	S 2	S 3	S 4	S 5	S 6
Web-Gaming Ratio	web	web	balanced	balanced	gaming	gaming
Low-High Gaming Ratio	low	high	low	high	low	high
Web	70%	70%	50%	50%	30%	30%
Low	16.7%	12.3%	27.85%	20.4%	39%	28.6%
Medium	10.2%	10.1%	17.05%	16.9%	23.9%	23.7%
High	3.1%	7.6%	5.1%	12.7%	7.1%	17.7%

4.3.5 Second Test Set

The second test was designed to test the overall ability of different algorithms to provision resources effectively in a realistic internet cafe scenario. Larger internet cafes are open 24 hours, 7 days a week and for this reason, the second test set takes place over an entire day of operation. The test set starts at the least busy hour and ends 24 hours later allowing the system to warm up and warm down around the busiest period. For these problems, users arrive in 15-minute intervals, a short enough time scale to be realistic for real-world arrivals.

After using test set one to test resource efficiency, further testing was carried out with test set two. This necessitates a change in service requirements and VM specifications to better match the desired performance for a more realistic and modern data set. The additional change to arrival times from every hour to every 15 minutes and total time period to 24 hours further improve the data sets realism.

This test set was used to test three usage profiles: 1) realistic, 2) stress, and 3) a special example case, these were tested with three scenario sets. The realistic set has internet cafes of real world size and with similar user profiles to real world internet cafes. The stress set has a large internet cafe with user profiles exceeding the design capacity of the internet cafe. The corner case or special set is a smaller test set designed to illustrate potential limitations of algorithms when presented with difficult but realistic examples of user demand.

Testing the effects of internet cafe size on these algorithms is as important as it was in test set one. Three sizes of internet cafes are considered in the realistic set: 120, 250, and 500

seats. The average internet cafe size in China is 120 seats, as such this test set varies from average to large.

In addition to the realistic scenarios, the stress set also uses the 500 seat internet cafe but extends the average length of stay for users to test when the internet cafe is significantly over maximum capacity. This is designed as a stress test for algorithms.

The final set is the special test which is built on an internet cafe with only 25 seats and a single set of users to show the potential pitfalls of no rejection algorithms.

These three tests all utilise the same base specifications defined next in §4.3.6 and §4.3.6.1.

4.3.6 Services offered

The services offered in the second test set are; web; low; medium; and high as in test set one. These levels are as described in §4.3.

Prices for test case two are based on realistic pricing for zoned internet cafes in China. These prices are no longer proportional to the resources required for the service. Prices and requirements are stated in table 4.7. The service requirements in test set two are adjusted from the test set one as the 3DMark scores required have changed over time so specific game testing was completed. The second test set has the service levels defined by specific games with minimum resolutions, frame-rates and settings. All these checks were done for VMs running Windows 10.

TABLE 4.7: Profit and performance details for test set two services

Service	Hourly Price	Games	FPS	Resolution	Settings
Web	\$ 0.80	None	N/A	1080p	N/A
Low	\$ 1.20	League of Legends, DoTA2, Hearthstone, World of Warcraft	40	1080p	medium
Medium	\$ 1.60	Overwatch, Warframe	40	1080p	medium
High	\$ 2.00	Witcher 3, Fallout 4	30	1080p	medium

4.3.6.1 Server Inputs

In this test set, there are two types of servers which supply VMs for services. These are the servers described in §3.1.1 and are the same as in test set one. These servers have either a K1 GPU or a K2 GPU card, again called the K1 server or K2 server respectively. These servers each have 32 CPU cores and 65GB of RAM. The quantity of RAM has changed from test set one to match the usage rates with that of the CPU cores. The number of servers used to provide services to users is described in table 4.8 where the total number of seats determines the quantity of servers. These server numbers are determined such that the internet cafe can supply all seats in the internet cafe with VMs if demand is average.

TABLE 4.8: Test set two Internet Cafe Sizes and Server Quantities

Size	Total Users	Seats	Number of K1 Servers	Number of K2 Servers
Special	30	25	0	4
Small	300	120	4	10
Medium	625	250	8	22
Large	1250	500	16	44

Servers must run VMs which meet the minimum performance requirements for each service even if the server resources are at capacity. Table 4.9 shows the specifications of VMs required to supply services on the K1 and K2 servers. The K1 server is unable to host medium or high gaming VMs. Each web VM supplies 4 users with each user effectively using 2 cores each. Testing found that Windows 10 performance was sluggish when run on a VM that was allocated only 2 of a server's CPU cores. However, even when allocated extra cores the VM does not use over 2 CPU cores except when opening applications and for initial Windows startup. The extra cores are needed to provide extra processing capacity in those small intervals. For this reason, web VMs have 8 cores shared between 4 users.

In this test set, it is not possible to mix and match VMs in a way that breaks vGPU rules as they either use an entire core or the same vGPU type. Configurations are not needed as they are quantities of VMs placed on a single GPU core used to stop cores from switching between vGPU types. Since this is not possible in this test, it was not necessary to build configurations. Instead the VMs are treated individually.

TABLE 4.9: Test set two virtual machine specifications

		K1 Server				K2 Server			
Service	Users Per VM	CPU	RAM	vGPU	GPU %	CPU	RAM	vGPU	GPU %
Web	4	8	16	none	0	8	16	none	0
Low	1	6	8	K180	1	4	6	K240	0.25
Medium	1	N/A	N/A	N/A	N/A	6	8	K240	0.5
High	1	N/A	N/A	N/A	N/A	8	8	K280	1

4.3.6.2 User Inputs

Users for this demand set require an arrival time, a number of hours to stay, and a service to demand. As with test set one no hard data was available on the exact usage patterns of internet cafe users. Fortunately, plenty of surveys exist on the habits of users as discussed in §2.1. For this test set, a survey of internet cafe users in China was used alongside surveys used in test set one to build a realistic representative data set.

This survey divided user arrival times into six times of day: early morning, morning, midday, afternoon, evening, and late night. Few users used the internet cafe in the early morning, and morning, with the quantity increasing throughout the day, rising in the afternoon into the evening and lasting until late night. The probability that any particular user arrives at any hour is shown in figure 4.3. These users are then uniformly allocated a 15-minute interval within that hour to arrive.

The survey showed around 30% of users would demand the web service, with the remaining playing games. However, this survey did not specify the games being played.

To generate gaming service demand of either low, medium or high, reports from AMD’s Raptr software from November 2015 were used (more recent than the reports for test set one) [141]. AMD Raptr is driver management software for AMD GPUs. The games in this list were divided into low, medium, and high based on their recommended hardware. The low service was set as requiring a GPU from Nvidia’s 200 series card or lower. The medium service was set as requiring a GPU between Nvidia’s 600 series and 300 series. The high service was set as requiring a GPU between Nvidia’s 900 series and 700 series. This categorisation leads

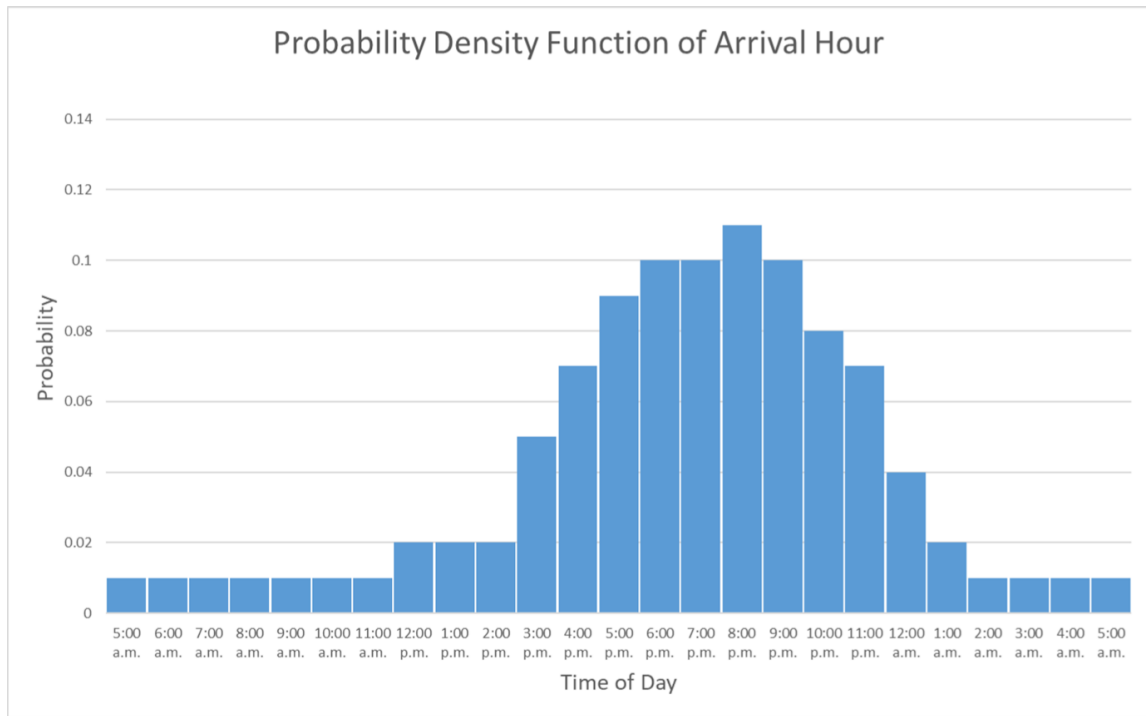


FIGURE 4.3: Probability density function for hours users arrive in test set two

to the top 20 most played games split: 70% at low, 15% at medium, and 15% at high. It is likely internet cafes only install around 20 games on their systems at any point in time. These games would likely be the 20 most popular games, so the distribution of the top 20 is used for the test set.

Combining these web demands and gaming demands creates the demand proportions shown in table 4.10.

TABLE 4.10: Proportion of demand for each service for test set two

Service	Percentage Demanding
Web	30.0%
Low	49.0%
Medium	10.5%
High	10.5%

The survey found that the average user stays 2 and a half hours before leaving. A distribution of hours stayed was created using a gamma distribution shown in figure 4.4 which roughly reflects the hours' users choose to stay.

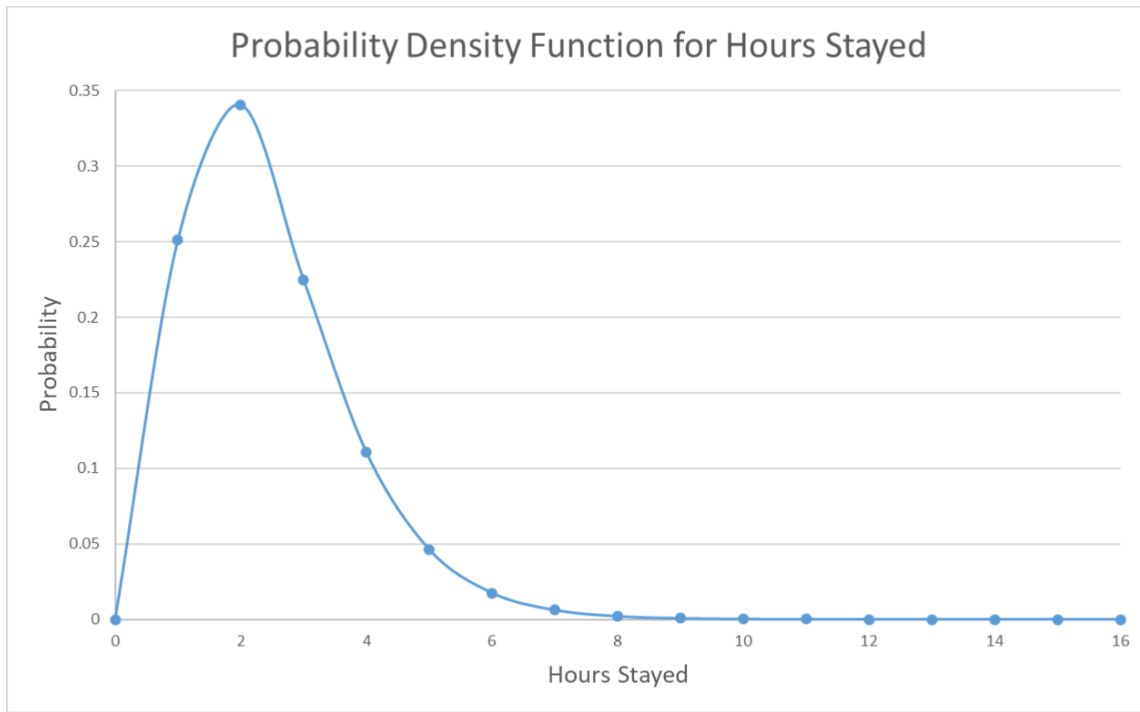


FIGURE 4.4: Probability density function for duration of stay in test set two for 2.5 hour average

It is necessary to create congestion in the internet cafe to test the algorithms in sub-optimal conditions for the stress test. To create congestion the time users stay is increased without changing the number of seats in the internet cafe. This increase was achieved by altering the gamma distributions and was done three times. First increasing the average stay time to 3 and a half hours as shown in figure 4.5. Second, the average stay was increased to 4 and a half hours as shown in figure 4.6. Lastly, the average was increased to 6 and a half hours which massively overloaded the internet cafe. This distribution is shown in figure 4.7.

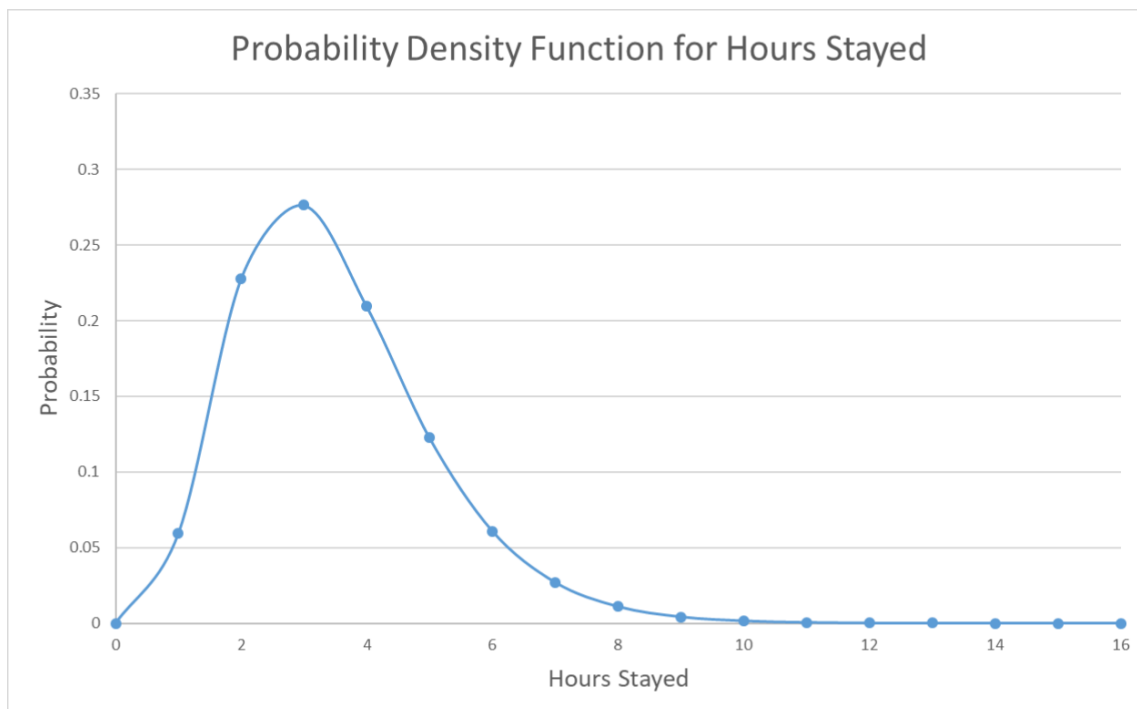


FIGURE 4.5: Probability density function for duration of stay in test set two for 3.5 hour average

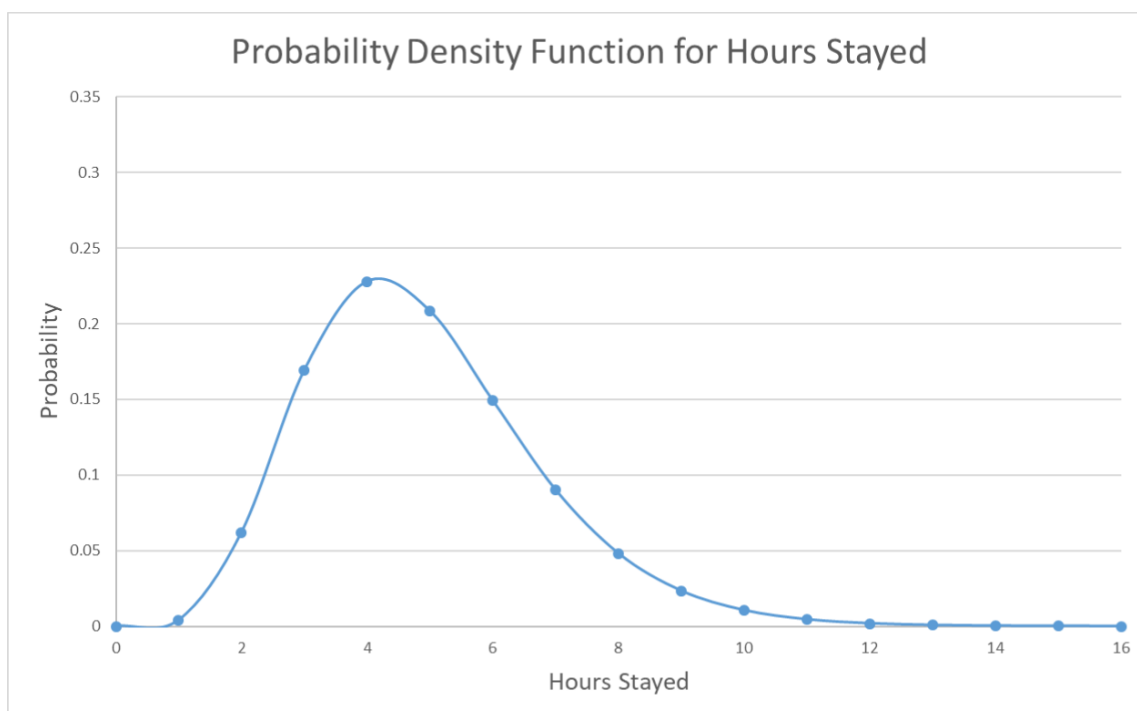


FIGURE 4.6: Probability density function for duration of stay in test set two for 4.5 hour average

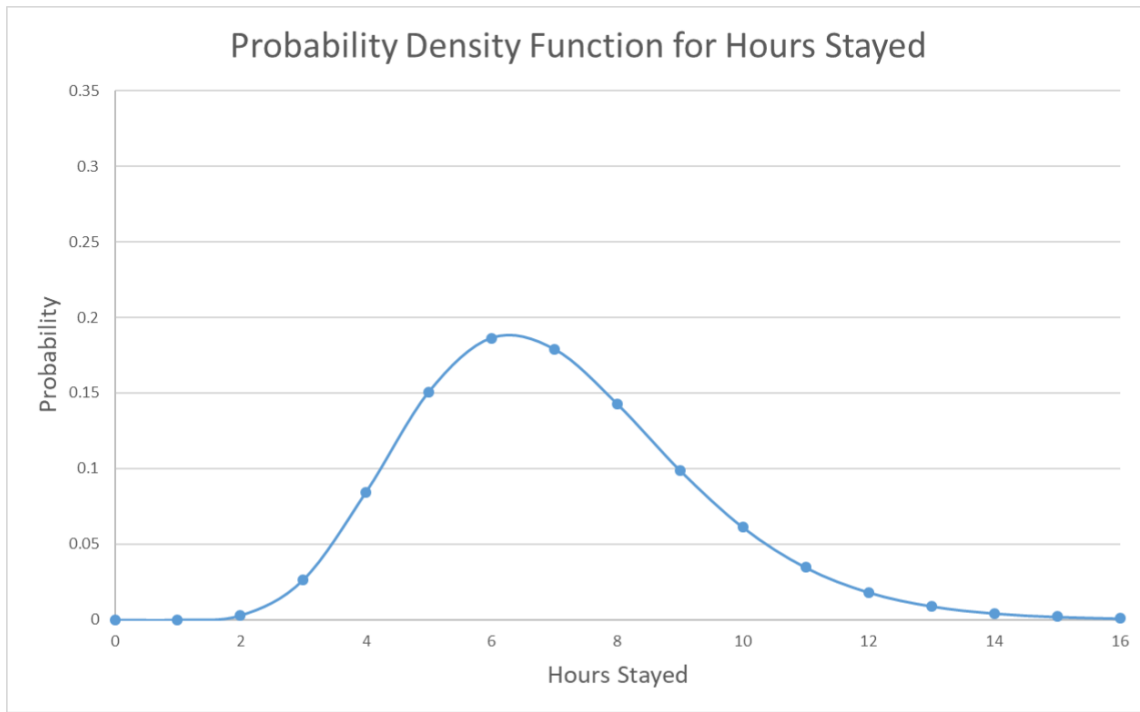


FIGURE 4.7: Probability density function for duration of stay in test set two for 6.5 hour average

A final special case was created using the same servers, VMs and services but with special users. The number of servers is shown for the 25 seat example in table 4.8. The users for the special test case consist of a set of exactly 20 students arriving in the internet cafe at 3 pm and staying until dinner time at 6 pm, and a set of exactly 10 workers arriving at the internet cafe at 5 pm and staying until 10 pm. This special case is designed to show the pitfalls of always accepting users as the students will fill the seats in the internet cafe leaving no room for the more profitable longer staying workers later in the day. Additionally, as there are only 10 workers, it is important to still accept some students as the workers will only use half the available resources. There was no randomness in this test case, all values were fixed for a single test.

4.3.7 Generation Process

For each problem 10 random realisations are generated for the given number of users. In the generation process each user is assigned:

- 1) An arrival time (first allocated an hour in the day followed by the specific time within the hour) generated from the distributions shown in figures 4.1, and 4.3.
- 2) A demand for a service which will be generated randomly from the proportions shown in tables 4.6, and 4.10.
- 3) A total number of hours the user will stay, generated from the gamma distributions shown in figures 4.2, 4.4, 4.5, 4.6, and 4.7.

4.3.8 Test Profiles Summary

Overall there are four distinct test sets generated to benchmark the algorithms.

The first test set benchmarks the potential improvement in resource efficiency that may come from a cloud-based internet cafe. It consists of a 10 hour period with users arriving every hour. It tests three different sized internet cafes with six unique service demand profiles. User stay duration's are generated from the same distribution for all sizes and demand profiles with an average stay of two and a half hours.

The second test set benchmarks: 1) the overall potential of algorithms in a real cloud-based internet cafe; 2) the efficacy of the algorithms as demand increases; and 3) the advantages of algorithms that include rejection. For realism, the test consists of a full day cycle with users arriving every 15 minutes. The realistic test contains three different sized internet cafes with a single service demand profile. User stay duration's are generated from the same distribution with the average user staying 2.5 hours. For the stress test with increasing demand, the largest internet cafe was utilised and user stay duration's were increased to provide 3 tests with increasing stays averaging 3.5, 4.5, and 6.5 hours. For testing rejection, a special test was designed to show the advantages of an algorithm with rejection over an accept all policy. This set has a small 25 seat internet cafe where users arrive after school and then again after work before the first set of users leave. Overall, the second test builds a profile for benchmarking algorithms against one another to operate a cloud-based internet cafe efficiently.

Chapter 5

Offline Integer Program

This chapter describes the different integer programs that were developed to solve the cloud-based internet cafe resource allocation problem. These solve the offline problem where all information is known when the problem is solved. The first integer program developed handles the rules for [vGPUs](#) described in [§2.2.5](#). The model is described in [§5.1](#). This model proved difficult to solve, and alterations were made to improve solution time and quality.

A second model was also developed which assumes [vGPU](#) rules will not be violated due to the natural structure of the data set is described in [§5.3](#). This model is paired with a third model which calculates the optimal zone size for a traditional zone-based non-cloud internet cafe described in [§5.4](#).

Finally, these models are used to solve the test data sets described in [§4.3](#) and the full results are presented.

These models are built from the problem description, inputs, and data sets described in [Chapter 4](#).

5.1 Initial Model

The integer program in this section was developed to solve the resource allocation problem for a cloud-based internet cafe. The decision variables, objective, and constraints are all fully described in this section.

This model was presented at the Utility and Cloud Computing Conference [142]. The offline model presented here was designed to comply with all known restrictions in the design of a cloud-based internet cafe. The model assumes that a sufficiently large and powerful remote storage block is available and that all connections are not limited by the network.

The model is built to place users on servers and GPU cores. It does this by setting server GPU cores into specific configurations which run VMs that can supply a user with their desired service. It attempts to set these configurations such that maximum profit is obtained from users accepted into the internet cafe. The models are constructed from the problem description in §4.1 and using model data described in §4.2.

5.1.1 Decision Variables

The model has the following binary decision variables:

For each user $u \in U$, server $s \in S$, and every GPU core $k \in K(s)$ of server s , there is a binary variable $x_{u,s,k}$ indicating if u is allocated to GPU core k :

$$x_{u,s,k} = \begin{cases} 1 & \text{if user } u \text{ is allocated to core } k \text{ of server } s \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

Similarly, for each user $u \in U$ and server $s \in S$, there is a binary variable $w_{u,s}$ indicating whether u is allocated to server s without assigning the user to a GPU core:

$$w_{u,s} = \begin{cases} 1 & \text{if user } u \text{ is allocated to server } s \text{ directly} \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

Constraints presented in §5.1.2 ensure that only gaming users are assigned to GPU cores and only web users are assigned directly to servers.

To deal with the configurations of GPU cores, a binary variable $y_{s,k,m,e}$ is used for each server $s \in S$, GPU core $k \in K(s)$, configuration $m \in M$, and event point $e \in E$ to indicate whether GPU core k is in configuration m at event point e :

$$y_{s,k,m,e} = \begin{cases} 1 & \text{if core } k \text{ of server } s \text{ is in configuration } m \\ & \text{at event point } e \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

Similarly, a binary variable $z_{s,m,e}$ is used for each server $s \in S$, configuration $m \in M$, and event point $e \in E$ to indicate whether the server itself is in configuration m at event point e :

$$z_{s,m,e} = \begin{cases} 1 & \text{if server } s \text{ is in configuration } m \text{ at event point } e \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

Constraints presented in §5.1.2 will ensure that only configurations $m \in M$ with $g(m) = \text{none}$ are active on servers, and only configurations with a matching type of GPU are active on each GPU core.

To model when a GPU core changes to a configuration with a different type of vGPU than

before (which is only possible if no users are currently assigned to the core), a binary variable $v_{s,k,e}$ is used for each server $s \in S$, GPU core $k \in K(s)$, and event point $e \in E$:

$$v_{s,k,e} = \begin{cases} 1 & \text{if core } k \text{ of server } s \text{ changes its configuration} \\ & \text{to a different } \zeta \text{ (vGPU) at event point } e \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

5.1.2 Integer Program

The objective is to maximise the total profit obtained by accepting users (profit of a user multiplied by whether or not they are allocated to a server directly – $w_{u,s}$ – or allocated to a GPU core – $x_{u,s,k}$):

$$\max \sum_{u \in U} \left(p(u) \sum_{s \in S} \left(w_{u,s} + \sum_{k \in K(s)} x_{u,s,k} \right) \right) \quad (5.6)$$

In order to obtain a feasible assignment of users to servers/cores, the constraints in §5.1.2.1-5.1.2.3 are used:

5.1.2.1 User constraints

Each user $u \in U$ can only be allocated once:

$$\sum_{s \in S} \left(w_{u,s} + \sum_{k \in K(s)} x_{u,s,k} \right) \leq 1 \quad \forall u \in U \quad (5.7)$$

The internet cafe must have enough seats for all users at any point $e \in E$:

$$\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}} \sum_{s \in S} \left(\sum_{k \in K(s)} x_{u,s,k} + w_{u,s} \right) \leq \bar{N} \quad \forall e \in E \quad (5.8)$$

The constraint counts the number of users active in each event point (i.e. when the point is after the user arrives $a(u)$ but before they leave $d(u)$) and ensures the total active users is less than or equal to the number of available seats.

5.1.2.2 Server constraints

Each GPU core $k \in K(s)$ of a server $s \in S$ can only be in one configuration at each event point $e \in E$:

$$\sum_{m \in M} y_{s,k,m,e} \leq 1 \quad \forall s \in S, k \in K(s), e \in E \quad (5.9)$$

Each server $s \in S$ itself can only be in one configuration at each event point $e \in E$:

$$\sum_{m \in M} z_{s,m,e} \leq 1 \quad \forall s \in S, e \in E \quad (5.10)$$

The GPU cores $k \in K(s)$ of each server $s \in S$ can only run configurations whose type of GPU matches their GPU type:

$$\sum_{e \in E} \sum_{k \in K(s)} \sum_{\substack{m \in M \\ g(m) \neq g(s)}} y_{s,k,m,e} = 0 \quad \forall s \in S \quad (5.11)$$

Configurations $m \in M$ that require a GPU core (i.e., with $\zeta \neq \text{none}$) can not be active on the servers directly (i.e. without allocating a GPU core):

$$\sum_{e \in E} \sum_{s \in S} \sum_{\substack{m \in M \\ \zeta(m) \neq \text{none}}} z_{s,m,e} = 0 \quad (5.12)$$

The RAM of each server $s \in S$ must not be over-allocated at any event point $e \in E$:

$$\sum_{m \in M} \rho(m, \text{ram}) \left(z_{s,m,e} + \sum_{k \in K(s)} y_{s,k,m,e} \right) \leq c \quad \forall s \in S, e \in E \quad (5.13)$$

The CPU cores of each server $s \in S$ must not be over-allocated at any event point $e \in E$:

$$\sum_{m \in M} \rho(m, \text{cpu}) \left(z_{s,m,e} + \sum_{k \in K(s)} y_{s,k,m,e} \right) \leq c(s, \text{cpu}) \quad \forall s \in S, e \in E \quad (5.14)$$

5.1.2.3 Event constraints

The supply of services (i.e. running configurations with VMs for those requirements) on each server and GPU core on that server – represented on the right-hand side of (5.15) – must satisfy the demand by active users at each event point – represented on the left-hand side of (5.15). Each GPU core $k \in K(s)$ of a server $s \in S$ must be able to supply the services of each requirement type $r \in R$ which are demanded by the users allocated to the core at each event point $e \in E$:

$$\sum_{\substack{u \in U \\ r(u)=r \\ a(u) \leq e < d(u)}} x_{u,s,k} \leq \sum_{m \in M} v(r, m) \cdot y_{s,k,m,e} \quad \forall s \in S, k \in K(s), r \in R, e \in E \quad (5.15)$$

Note that at most one of the variables $y_{s,k,m,e}$ in the sum on the right hand side can be equal to one. Also note, since $v(\text{web}, m) = 0$ for all configurations $m \in M$ that are feasible for GPU cores (i.e. have $\zeta(m) \neq \text{none}$), this constraint also ensures that only gaming users are assigned to GPU cores.

A second constraint is set up, similar to (5.15), for services which do not require a GPU core. Each server $s \in S$ must be capable of supplying the services of each requirement type $r \in R$ demanded by the users allocated to the server directly at each event point $e \in E$:

$$\sum_{\substack{u \in U \\ r(u)=r \\ a(u) \leq e < d(u)}} w_{u,s} \leq \sum_{m \in M} v(r, m) \cdot z_{s,m,e} \quad \forall s \in S, r \in R, e \in E \quad (5.16)$$

Note that, since $v(r, m) = 0$ for all $r \neq \text{web}$ and all configurations $m \in M$ that are feasible for being active on a server directly (i.e. have $\zeta(m) = \text{none}$), this constraint also ensures that

only web users are assigned directly to servers.

If, at event point $e \in E$, a GPU core $k \in K(s)$ of a server $s \in S$ changes from a configuration $m' \in M$ that was active on the core at the previous event point e^- to a configuration $m \in M$ having a different type of vGPU, the variable $v_{s,k,e}$ must be set to one:

$$1 + v_{s,k,e} \geq y_{s,k,m,e} + \sum_{\substack{m' \in M \\ \zeta(m') \neq \zeta(m)}} y_{s,k,m',e^-} \quad \forall s \in S, k \in K(s), m \in M, e \in E \quad (5.17)$$

Observe that at most one of the variables $y_{s,k,m',e}$ in the sum on the right hand side can be equal to one.

A core $k \in K(s)$ of a server $s \in S$ cannot change to a configuration with a different type of vGPU at event point $e \in E$ if a user $u \in U$ is allocated to the core at event point e :

$$1 - x_{u,k,s} \geq v_{s,k,e} \quad \forall s \in S, k \in K(s), u \in U, e \in E : a(u) < e < d(u) \quad (5.18)$$

5.2 Efficiency Improvements

The model was difficult to solve due to symmetry arising from homogeneous servers. There were also memory issues due to the large number of variables and constraints. Epsilon objective perturbations and constraint branching were implemented to help with the symmetry problem. Variables and constraints which would be set to 0 were instead removed, and lazy constraints were utilised to reduce the memory usage. This section discusses each of these problems and subsequent approaches in more detail.

5.2.1 Symmetry

The inherent problem symmetry made the model difficult to solve. As there were only two types of servers in the test data sets, it was possible to swap users between servers without changing the objective value while creating distinct solutions. This symmetry prevented the

branch and bound tree from fathoming nodes. To differentiate solutions from one another the cost coefficients were altered with an epsilon objective perturbation. The epsilon perturbation randomly perturbs the cost coefficients in an attempt to remove symmetric solutions with equal objective function values. Essentially each user, server pairing was given a small unique coefficient alteration. These perturbations created a slight difference in objective when placing different users on different servers allowing the branch and bound tree run more efficiently.

The new objective used was:

$$\max \sum_{u \in U} \left(p(u) \sum_{s \in S} \left(w_{u,s} + \sum_{k \in K(s)} x_{u,s,k} \right) \right) + \sum_{u \in U} \left(\sum_{s \in S} \epsilon \cdot (\text{index}(u) + \max(\text{index}(s))) \cdot \text{index}(s) \left(w_{u,s} + \sum_{k \in K(s)} k \cdot x_{u,s,k} \right) \right),$$

where ϵ was a small value which didn't change the optimal solution from the formulation without the perturbation. For this problem, a value of $\epsilon = 10^{-5}$ was used. The ϵ value was determined by the largest index value generated by the equation compared to the smallest possible objective profit. The largest index value is calculated from the largest user index of 1000 and the largest server index of 50 giving a maximum combined index on the order of 10^4 . The lowest profit would be the cheapest user staying for one hour with a profit of 1. As such the epsilon value of 10^{-5} ensures the order of the perturbation is always smaller than the objective gained when accepting any user, meaning a user will not be accepted simply for having a high user and server index. Overall, epsilon constraint perturbations enabled high quality solutions to be found faster. When the solver was given 10 minutes to find solutions the bounds were improved from 10% to under 1% for many problem instances.

A second technique that was utilised in order to (attempt to) reduce symmetry was constraint branching. Constraint branching branches on constraints, effectively the sum of a set of variables, before branching on variables directly. In this case, new variables were created which represented the total number of users of each service type on a server in each time period. Constraint branching then branches on the number of users of each type rather than

the individual users. These additional variables and constraints increased the build time of the model and increased the solution time for the linear program relaxation. This time increase was significant enough that the improvement from constraint branching was entirely negated.

Column generation techniques including Dantzig-Wolfe decomposition[143] were also considered to reduce symmetry because these methods have been shown to reduce symmetry in bin packing problems. Column generation was tested using Dippy [144] which has an automatic column generation function but the CBC solver used was unable to handle such large problem sizes. A manual reformulation was also considered, however, the epsilon constraint perturbations had already sufficiently reduced symmetry for all practical use cases with the longest solve times needed at only around 10 hours.

5.2.2 Memory Use

Given the size of the real world problem instances, memory requirements were also a solution issue. For initial data sets memory was not a problem, but when the problem size was significantly increased the memory required became a problem. The increased memory requirement arose from the data sets with 15 minute time intervals over 24 hours versus the previous hourly intervals. This finer discretisation of time resulted in millions more constraints and variables. The first solution to this problem was the removal of variables which were directly set to 0 by the constraints. Removing these variables also allowed the removal of constraints limiting these variable values. The removal of these variables and constraints reduced the total memory requirement by around 40%.

The variables removed in this process were those for servers upon which a user could never be placed (i.e., the server did not have a [VM](#) capable of the demanded service). In addition, variables were defined and then set to zero for allocating gaming users directly to the server when they required a [GPU](#) core. These were also removed along with the related constraints.

5.3 Updated Model

When solving the cloud-based internet cafe resource allocation problem for test data set 2 (as described in §4.3.5), many of the vGPU complexities were not required due to the structure of test set 2. The reduced vGPU complexity meant it was no longer necessary to track vGPU types and prevent the switching of vGPU types, as this can not happen in test set 2. For this reason a new faster model was built to solve test set 2 efficiently.

The online models in chapter 7 required assumptions about GPU virtualisation which are met by the 24-hour data sets. These assumptions are that: there is no scenario where the type of vGPU will change infeasibly (i.e., from a K260 type to a K240 while a VM is running), and all GPU cores can be treated as continuous resources like CPU and RAM (i.e., all combinations of VMs will add to whole numbers of GPU cores). If these assumptions hold true, then the variables for changing configurations are no longer needed, in fact, configurations themselves are no longer needed, and it is instead possible to track individual VMs. This model has the added advantage of scaling better to the 15 minute time intervals used for test set 2 presented in §4.3.5.

Both the prebooking integer program presented in chapter 6 and the offline algorithm comparisons for the online models in chapter 7 make use of this model.

5.3.1 Decision Variables

The model uses the following binary decision variables:

Similar to the previous formulation in (5.1) and (5.2), for each user $u \in U$, server $s \in S$, and every GPU core $k \in K(s)$ of server s , there is a binary variable $x_{u,s,k}$ indicating whether u is allocated to GPU core k on server s :

$$x_{u,s,k} = \begin{cases} 1 & \text{if user } u \text{ is allocated to core } k \text{ of server } s \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

Due to the lack of vGPU limitations this formulation is able to consider individual VMs instead of configurations. Each non-web user represents a single VM but the same is not true for web users. Therefore, for each server $s \in S$, and event point $e \in E$, there is an integer variable $y_{s,e}$ indicating the number of web virtual machines being run on server s :

$$y_{s,e} \geq 0 \quad (5.3')$$

Here we use the dummy core $k = 0$ to represent placing web users directly onto the server without using GPU resources and extend $K(s)$ (and the definition of x) to include a dummy core. As web users do not require GPU resources, all variables for placing web users onto GPU cores are set to 0 and removed in pre-processing as described in §5.2.2:

$$x_{u,s,k} = 0 \quad \forall u \in U, s \in S, k \in K(s) : r(u) = \text{web}, k \neq 0 \quad (5.19)$$

All non-web users require a GPU core, and as such must be allocated to a non-dummy GPU core. All variables relating to the dummy GPU core ($k = 0$) are set to 0 and removed in pre-processing as above:

$$x_{u,s,0} = 0 \quad \forall u \in U, s \in S : r(u) \neq \text{web} \quad (5.20)$$

Users can not be placed on servers to which they cannot be assigned (these variables are also removed in pre-processing):

$$x_{u,s,k} = 0 \quad \forall u \in U, s \in S \setminus z(u), k \in K(s) \quad (5.21)$$

5.3.2 Integer Program

The objective is to maximise the total profit obtained by accepting users. Note that this is the same objective as (5.6), but $w_{u,s}$ has been replaced by $x_{u,s,0}$ with $K(s)$ has been extended

to include the dummy core $k = 0$. The objective definition is:

$$\max \sum_{u \in U} \left(p(u) \sum_{s \in S} \left(\sum_{k \in K(s)} x_{u,s,k} \right) \right) \quad (5.6')$$

Note the previous constraints, (5.19)- (5.21), ensure that invalid x variables are set to 0.

In order to obtain a feasible assignment of users to servers/cores, the constraints in §5.3.2.1 and §5.3.2.2 are used:

5.3.2.1 User Constraints

Each user $u \in U$ can only be allocated at most once (mirroring (5.7)):

$$\sum_{s \in S} \left(\sum_{k \in K(s)} x_{u,s,k} \right) \leq 1 \quad \forall u \in U \quad (5.7)$$

The internet cafe must have enough seats for all users at any point $e \in E$ (mirroring (5.8)):

$$\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}} \sum_{s \in S} \sum_{k \in K(s)} x_{u,s,k} \leq \bar{N} \quad \forall e \in E \quad (5.8)$$

5.3.2.2 Supply Constraints

Enough virtual machines for the number of web users allocated to the server must be supplied (this constraints mirrors (5.16) but is formulated differently due to the removal of configurations from the problem):

$$\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}} x_{u,s,0} \leq \bar{n}(s, \text{web}) y_{s,e} \quad \forall s \in S, e \in E \quad ((5.16)')$$

Note that y is multiplied by \bar{n} as y represents the number of non-GPU VMs on s and each of these VMs can supply multiple users (equal to \bar{n}).

Each server must have enough **RAM** and **CPU** to supply the virtual machines at any point $e \in E$ (mirroring constraints (5.14) and (5.13) in the initial model):

$$\underbrace{\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}}}_{\text{sum over non-web users active at } e} \left(\underbrace{b(s, r(u), q) \sum_{\substack{k \in K(s) \\ k \neq 0}} x_{u,s,k}}_{q \text{ required for non-web user } u \text{ on } s} \right) + \underbrace{b(s, \text{web}, q) y_{s,e}}_{\text{total } q \text{ for web users on } s} \leq c(s, q) \quad (5.22)$$

$$\forall s \in S, e \in E, q \in \{\text{cpu}, \text{ram}\}$$

This constraint calculates the resources consumed by each user active at each event point who has been allocated to the server and compares that against the resource available on a server. The users are split by users demanding **GPU VMs** (represented by x and those who do not (represented by the number of **VMs** y).

Each server must have enough **GPU** resources to supply the virtual machines at any point $e \in E$ (this is unique to the new formulation as previously the set up of the configurations ensured this was true):

$$\underbrace{\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}}}_{\text{sum over non-web users active at } e} \underbrace{(b(s, r(u), \text{gpu}) x_{u,s,k})}_{\text{GPU resource usage by users on server}} \leq 1 \quad \forall s \in S, k \in K(s), e \in E : k \neq 0 \quad (5.23)$$

Note that the capacity of each **GPU** core is 1 or 100%. Each user allocated to the **GPU** core utilises a percentage of the core while they are active.

5.4 Fixed Zone Model

In order to compare the initial and updated model results to a traditional internet cafe with zones for each service an integer program was created which calculates the optimal size of each zone to maximise profit. These zone sizes are fixed for the entire day. The profit can

then be compared to the profit of the cloud-based internet cafe model where zone sizes change throughout the day.

5.4.1 Decision Variable

The model uses the following binary decision variables:

As in the previous formulations from (5.1) and (5.2), for each user $u \in U$ there is a binary variable x_u indicating whether u is accepted:

$$x_u = \begin{cases} 1 & \text{if user } u \text{ is accepted} \\ 0 & \text{otherwise} \end{cases} \quad (5.1')$$

For each service $r \in R$ there is an integer variable y_r equal to the number of seats for service r in the internet cafe:

$$y_r \geq 0 \quad (5.3'')$$

5.4.2 Integer Program

The objective is to maximise the total profit obtained by accepting users:

$$\max \sum_{u \in U} p(u)x_u \quad (5.6')$$

The internet cafe must have enough seats for all users at any point $e \in E$:

$$\sum_{\substack{u \in U \\ a(u) \leq e < d(u)}} x_u \leq \bar{N} \quad \forall e \in E \quad (5.8')$$

The internet cafe must have enough seats for all users demanding $r \in R$ at any point $e \in E$.

This is the fixed zone internet cafe's equivalent of resource constraints in equations (5.14),

(5.13) and (5.22):

$$\sum_{\substack{u \in U \\ a(u) \leq e < d(u) \\ r(u)=r}} x_u \leq y_r \quad \forall e \in E \quad r \in R \quad (5.24)$$

The internet cafe must have enough seats for total seats over all zones:

$$\sum_{r \in R} y_r \leq \bar{N} \quad (5.25)$$

5.5 Results

These results were generated by solving the offline models in this chapter for the data sets described in §4.3. All models were built in Python 2.7 using PuLP [93] and solved using Gurobi [94].

5.5.1 Test Set One

These results are generated using the data outlined in §4.3.1 using the model outlined in §5.1. All six test cases in the data set were solved using Gurobi 6.0.3 through a VM running Windows 7 which had sixteen 2.1 GHz CPU cores and 16 GB of RAM. Gurobi [94] was used as the solver because solve time was a problem and Gurobi offered superior solve times compared to other solvers tested including IBM Cplex [97] and CBC [98]. Each test case had three sizes 100, 500, and 1000 users per day. Results presented are the average of 10 realisations (generated using the methodology in §4.3.7) of the random inputs in the test data set with 95% confidence intervals. Test set one consists of six scenarios each with different gaming demand levels ranging from low to high labelled 1 to 6 as described in table 4.6 (reproduced here). The results are compared to the profit obtained when running the fixed zone model on the same data.

TABLE 4.6: Proportion of demand for each service for test set one

	Percentage Demanding Service in Scenario					
	S 1	S 2	S 3	S 4	S 5	S 6
Web-Gaming Ratio	web	web	balanced	balanced	gaming	gaming
Low-High Gaming Ratio	low	high	low	high	low	high
Web	70%	70%	50%	50%	30%	30%
Low	16.7%	12.3%	27.85%	20.4%	39%	28.6%
Medium	10.2%	10.1%	17.05%	16.9%	23.9%	23.7%
High	3.1%	7.6%	5.1%	12.7%	7.1%	17.7%

The 100 user cases were able to reach optimality in a reasonable time period, as shown in table 5.1. For the cases with, 500 and 1000 users there was a slower rate of convergence to the optimal solution. For this reason, the model solve time was limited to ten minutes.

Tables 5.1, 5.2, and 5.3 show the gap between the final objective values and the theoretical upper bound as computed by the solver after ten minutes. All test cases reach an optimality gap below 5% within the ten-minute limit. Many models reach optimality or achieve a gap lower than 1%. The optimality gap generally increased as the number of users increased. The overall average gap after ten minutes was 1.5%.

TABLE 5.1: Comparison of results for test set one with 100 users after running each test for ten minutes

Scenario Number	1	2	3	4	5	6	Average
Number of Users	100	100	100	100	100	100	
Profit	327	324	382	387	432	431	0.28%
95% CI	± 3.5	± 3.3	± 4.8	± 4.7	± 4.5	± 5.7	
Optimality Gap	0.00%	0.00%	1.42%	0.26%	0.00%	0.00%	8.12%
Profit (Fixed Zone)	301	300	346	359	381	410	
95% CI	± 3.4	± 3.4	± 4.7	± 4.6	± 4.4	± 5.5	
Profit Improvement	7.95%	7.41%	9.42%	7.24%	11.8%	4.87%	

All test results were compared to the fixed zone version of the model which allowed only a single configuration of zoned seats within the internet cafe for the entire day. This version of the model was designed to correspond to a standard internet cafe with different zones of fixed sizes using an optimal user acceptance strategy. It is important to note that even with this fixed zone model the zone sizes are optimised for the specific demand given. For this reason, the fixed zone model obtains profit equivalent to the maximum profit achievable with fixed zone sizes for the given demand. In practice, the internet cafe would have to fix its zone sizes

over multiple days as demand changes, obtaining a lower profit than using zones that adjust to the days' unique demand.

This fixed zone model can be used to measure the merit of the increased flexibility offered by the cloud-based model where servers provide VMs that exactly meet the users demands at any point in time. Tables 5.1, 5.2, and 5.3 show that on average the flexible models offers a 5.5% improvement in profit over the restricted model with improvements as large as 11.8%. Additionally the initial model always produced a solution superior to that of the fixed model, showing the value of added flexibility. It is worth noting that increasing the solve time beyond 10 minutes could further increase this gap, although the remaining optimality gaps are smaller than the improvement currently offered by flexibility. This difference means the improvement demonstrated by the flexible model is predominantly due to the flexible approach providing better solutions since the optimality gap can only explain a small fraction of the improvements for the 100, 500, and 1000 user internet cafes. The flexible models have an average profit improvement of 8.12% for the 100 user internet cafe and a smaller 4.2% for the 500, and 1000 user internet cafes when compared with the fixed zone model. The smaller improvement in the larger problems may be because the improvement in profit from accepting an additional user in a 100 user internet cafe represents a larger percentage of the total profit than it does in either a 500 or 1000 user internet cafe.

TABLE 5.2: Comparison of results for test set one with 500 users after running each test for ten minutes

Scenario Number	1	2	3	4	5	6	Average
Number of Users	500	500	500	500	500	500	
Profit	1629	1782	1622	1920	1739	1995	
95% CI	± 17	± 18	± 22	± 27	± 21	± 23	
Optimality Gap	0.92%	2.97%	2.32%	1.32%	1.92%	0.87%	1.72%
Profit (Fixed Zone)	1595	1771	1517	1783	1648	1939	
95% CI	± 18	± 19	± 21	± 28	± 20	± 21	
Profit Improvement	2.09%	0.62%	6.47%	7.14%	5.23%	2.81%	4.06%

For all scenarios, the most interesting aspect is how resource utilisation is improved in comparison to the traditional internet cafe. Resource utilisation is the percentage of total available resources that are being used by users. Figure 5.1, and 5.2 show the percentage utilisation of

TABLE 5.3: Comparison of results for test set one with 1000 users after running each test for ten minutes

Scenario Number	1	2	3	4	5	6	Average
Number of Users	1000	1000	1000	1000	1000	1000	
Profit	3109	3519	3182	3800	3265	3763	2.55%
95% CI	± 41	± 41	± 32	± 52	± 44	± 52	
Optimality Gap	3.94%	3.01%	2.66%	1.16%	3.68%	0.86%	4.44%
Profit (Fixed Zoned)	3029	3469	2959	3496	3144	3614	
95% CI	± 40	± 42	± 33	± 50	± 42	± 51	
Profit Improvement	2.57%	1.42%	7.01%	8.00%	3.71%	3.96%	

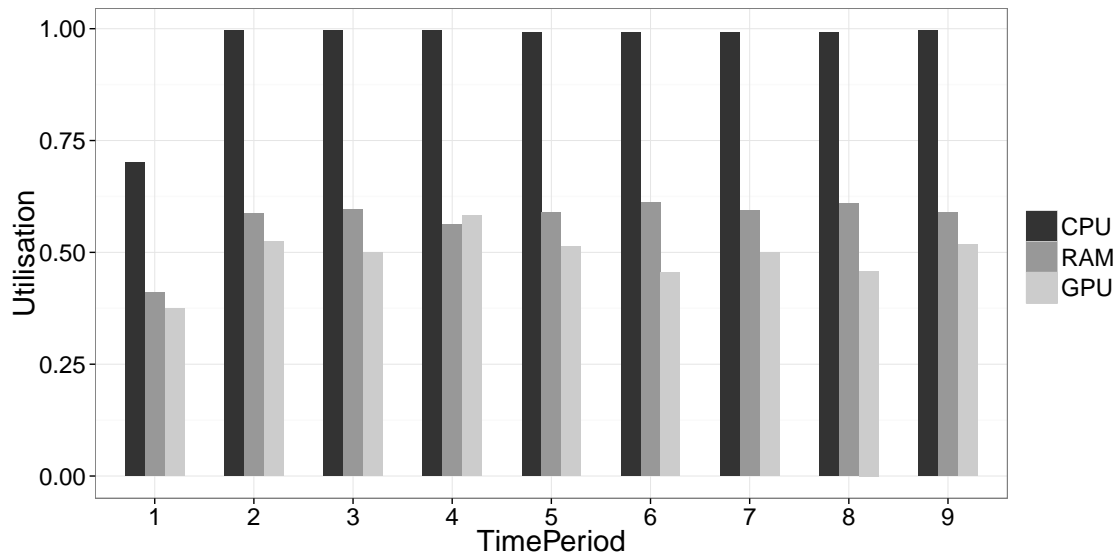


FIGURE 5.1: Utilisation of CPU, RAM, and GPU resources over time for test set one test 2

the CPU, RAM, and GPU resources on averaged over all servers for the 1000 user internet cafe in scenarios two and six respectively. A utilisation of 1 corresponds to 100% of the resource being used. The plots show CPU utilisation near 100% for the entire time period with the other resources around 50% utilised, compared to the lower utilisation of a traditional internet cafe as shown in figure 5.3. The under-utilisation of non-CPU resources could be avoided by increasing the number of CPU cores on the server or decreasing the total RAM. Unfortunately, GPU cards come in fixed quantities and are hard to adjust. One of the most significant contributors to low GPU resource utilisation are web users, who do not utilise any GPU consuming only CPU and RAM resources. The existence of web users may indicate a need for non-GPU servers to meet this demand exclusively.

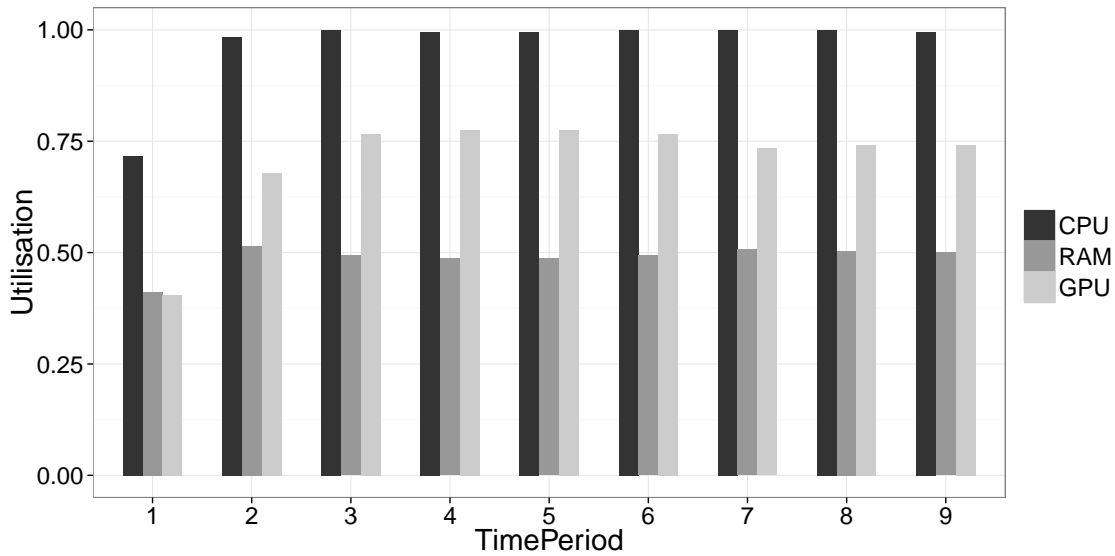


FIGURE 5.2: Utilisation of CPU, RAM, and GPU over time for test set one test 6

The results for scenario 6, the scenario with 70% gaming users, and a higher percentage of high gaming users is shown in Figure 5.2. A higher utilisation of GPU resources at around 70% is evident, as a result of scenario 6's higher gaming demand. However, this utilisation still doesn't reach 100% due to web users taking CPU cores without using any GPU resource. In fact, in test case 6 the K2 server GPU resource has a utilisation around 100% while the K1 server GPUs are only around 50% utilised. This lower K1 utilisation highlights a potential need for more K2 servers in this problem.

Figure 5.3 shows the resource utilisation of a standard internet cafe (where all seats provide all services) compared to the cloud-based internet cafe with both accepting the same users. In this standard internet cafe, all desktop machines have specifications that are the same as a high gaming VM, and utilisation is counted as the percentage of these resources used by the VM for the users' actual demanded service. The values clearly show the cloud-based model can offer significant improvements in CPU and RAM utilisation with a minor improvement in GPU utilisation. These results could be further improved if the CPU, RAM, and GPU resources were set to a more appropriate ratio. Nevertheless these results show how a cloud-based model can reduce the resource inefficiency found in a standard internet cafe.

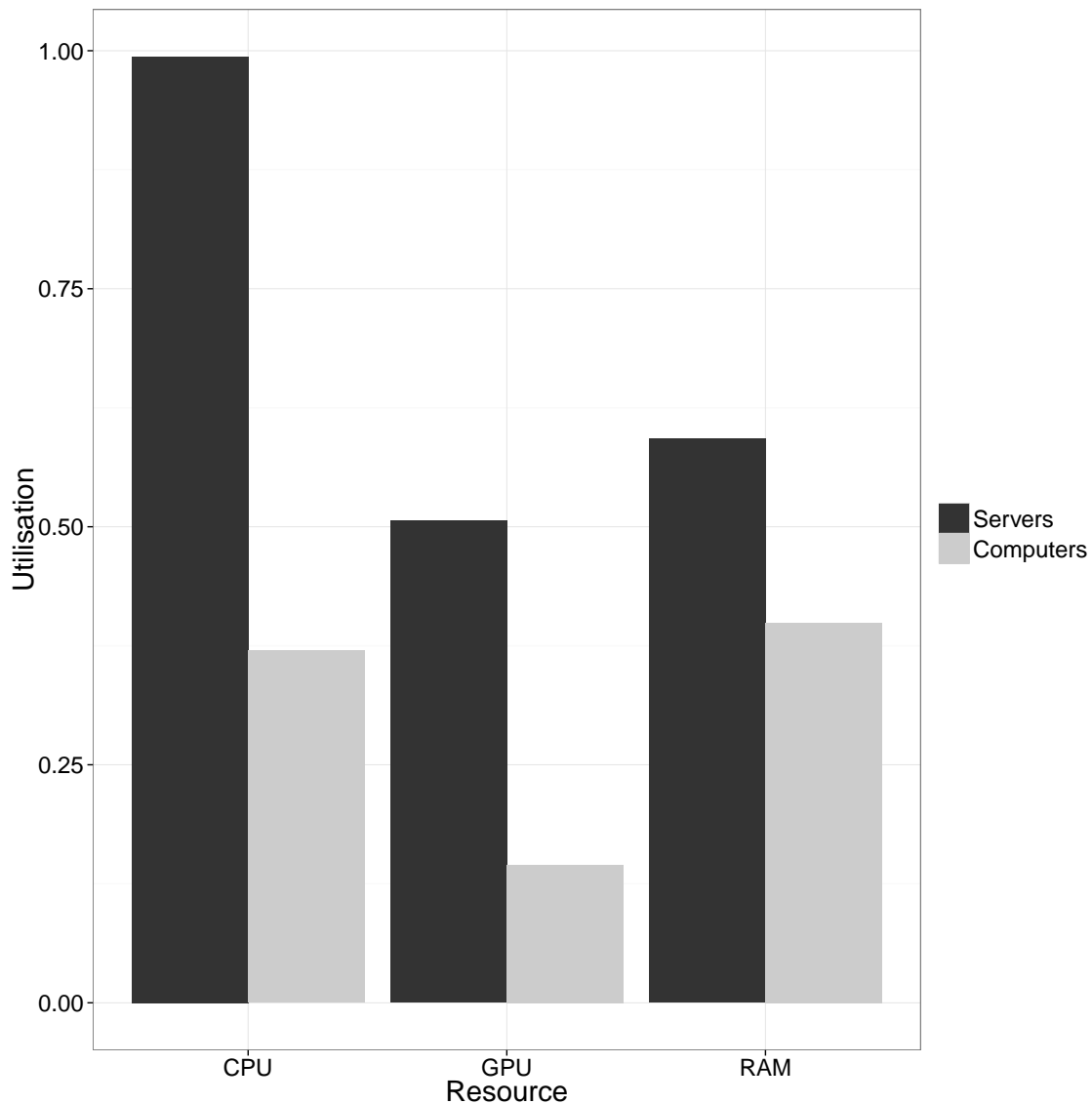


FIGURE 5.3: Comparison of average utilisation of resources using cloud-based vs. traditional internet cafe

5.5.2 Test Set Two

These results are generated using the data outlined in §4.3.5 with the model outlined in §5.3. All three test cases in the data set were solved using Gurobi 6.5.2 through a VM running Windows 10 which had sixteen 2.1 GHz CPU cores and 16 GB of RAM. Gurobi [94] was used as the solver as it had shown fast solve times in the previous test sets.

Results presented for each test case are the average of 10 realisations of the random inputs in the test data set, again generated using the methodology stated in §4.3.7.

Table 5.4 shows the profit and optimality gap obtained by the model after solving for 10 minutes and 10 hours. The realistic test set with 2 and a half hour stay times can solve to optimality within 10 minutes using the new model. The stress tests with longer stay duration's do not come close to optimality within the 10 minutes. In fact, for the 4.5-hour tests, only 5 of the 10 realisations even find integer solutions within 10 minutes. The other high capacity tests reach optimality gaps in the 20-30% range which is not a suitable solution.

TABLE 5.4: Comparison of results for test data set two with 10 minute solve times versus 10 hour solve times

Average Stay	Size	Profit (10 minute)	Optimality Gap	Profit (10 hours)	Optimality Gap	Change
2.5 Hours	300	1162.4	0%	1162.4	0%	0%
2.5 Hours	625	2429.1	0%	2429.1	0%	0%
2.5 Hours	1250	4895.9	0%	4895.9	0%	0%
3.5 Hours	1250	5137.4	22.7%	6068.4	0.95%	21.75%
4.5 Hours	1250	No Solution	NA	7184.7	2.22%	INF
6.5 Hours	1250	7337.2	28.4%	8754.5	1.5%	26.9%

The 10 hour solve time drastically improves the solution quality with the optimality gaps falling from over 20% down to below 2.5%. Even the 4.5-hour case reaches a 2.22% optimality gap despite not finding an integer solution within 10 minutes.

The model was also solved for the special 30 user test case presented as part of test set two in §4.3.5. This test case is used to show the power of user rejection when an internet cafe is over capacity with high profit users arriving after lower value users. The special test could be solved to optimality instantly with a profit of \$116.9. Accepting the 13 most profitable users who arrive at 3 pm and then filling the remaining server space with all 10 of the 5 pm arrivals. These results will be used later when comparing online algorithm placement performance in chapter 7. While this is a trivial model for the offline problem to solve to optimality, when presented with limited future information the solution to this problem varies drastically depending on the online strategy.

Chapter 6

Prebooking Integer Program

This chapter describes a prebooking system for a cloud-based internet cafe that can utilise the offline integer program from chapter 5 for placing users who have booked in advance.

In a prebooking system, users must request seats in the internet cafe in advance. Users must specify their desired arrival time, duration, and service. An algorithm decides to accept or reject users, and determines on which servers they will be placed. Users whose arrival time is within a defined window are notified whether their request has been accepted (this window is referred to as the “notification window”). The algorithm can be solved again as more user requests arrive. Users are progressively notified as their arrival enters the notification window. The advantage of a prebooking system lies in its ability to leverage future information. Future information allows the algorithm to make informed choices when accepting and placing users.

The prebooking system must provide an interface for users to book and a notification system to inform users of their request status. The backend of the system solves an algorithm that takes three input settings. These three inputs are: 1) δ (the solve frequency), the frequency with which the algorithm is run, 2) γ (the notification window), the quantity of time before arrival that users are informed of the outcome of their request, and 3) σ (the time horizon), the amount of future information the prebooking system is given. In order to meet the future information requirement from σ and the window for informing users of

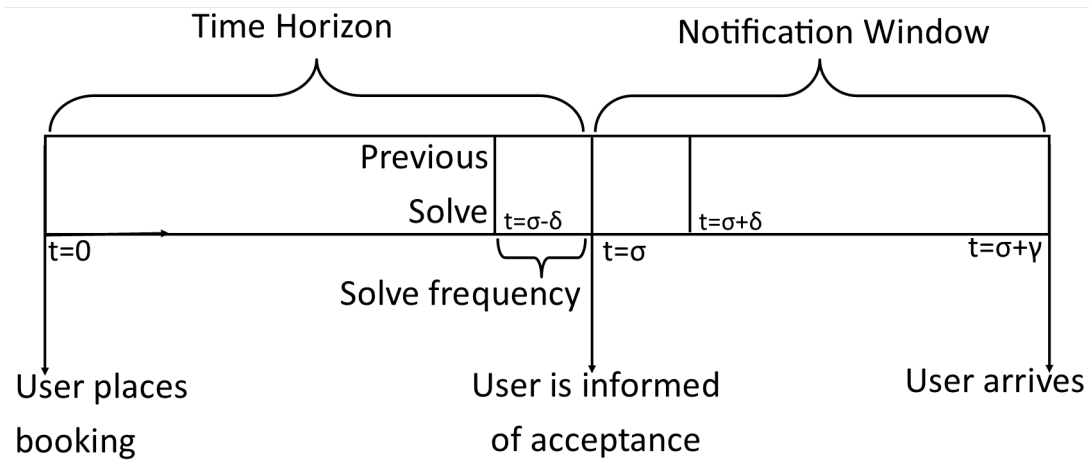


FIGURE 6.1: Features of a prebooking system over time from when a user places a booking

acceptance from γ , users must place bookings at the latest by $\sigma + \gamma$. This time period gives the prebooking system the necessary information to make its decision on user acceptance and inform users within the notification window. Note that users will have to wait σ units of time after placing their booking before they are informed of their acceptance or rejection. Figure 6.1 shows the system graphically, with a user placing a booking, waiting σ hours to be informed of acceptance or rejection and then finally arriving after $\sigma + \gamma$ hours since placing the booking.

The integer programs in chapter 5 are one of the best options for placement algorithms because, as shown in the previous chapter, they are able to produce provably optimal solutions for the problem formulated. These integer programs take the user requests and then decide whose bookings to accept and reject as well as which servers to place those bookings on. The user can then be notified of whether the request was accepted.

The prebooking system is tested using the data described in §4.3.5. The test steps through the data set from one user arrival time to the next in order to simulate users placing requests. Then the integer program's decisions are fixed as they enter the users notification window.

6.1 Model

The integer program presented in §5.3 is used to solve the user allocation for the prebooking system. This section presents a time-staged version of the IP which simulates the function of the prebooking system throughout the day.

To generate solutions for benchmarking that use an entire data set, it is necessary to simulate the placement of requests. This simulation is performed by iteratively solving the IP with subsets of the total data set based on the time horizon and solve frequency.

This time staged model represents the prebooking system placing users in the internet cafe over the course of a day. The simulation uses time horizon and solve frequency to decide how to subset test data set two. The time horizon σ defines the number of hours of future information available about bookings which have already been placed. The solve frequency δ defines how often new data is added, and the problem solution is recomputed.

6.1.1 Batch Solver

This section defines the selection of inputs for the batch solver IP from the inputs in §4.2.2. Consider δ to be the time between sequential solves and σ as the time into the future that user arrivals are known.

Using these values the first iteration of the integer program takes: event points $e \in E$ where $e \leq \sigma$ and users $u \in U$ where $a(u) \leq \sigma$ (i.e. users who have already placed bookings). Solving the integer program with these limited inputs gives a solution for the first iteration of user acceptance notifications.

Then stepping forward in time by δ gives the second iteration of the problem which includes: event points $e \in E$ where $\delta \leq e \leq \sigma + \delta$ and users $u \in U$ where $a(u) \leq \sigma + \delta$ and $d(u) > \delta$ (i.e. users who have placed bookings but are not yet within the notification window or users who have been notified but are still active for the times the model is considering).

Generalising gives the number of iterations in the batch solver as:

$$N = \left\lceil \frac{\max_{e \in E}(e)}{\delta} \right\rceil$$

For any problem instance $n \in 0 \dots N$:

- Select users: $U_n \subset U$ such that $a(u) \leq \sigma + n \cdot \delta$ and $d(u) > n \cdot \delta$
- Select event points: $E_n \subset E$ such that $n \cdot \delta \leq e \leq \sigma + n \cdot \delta$
- For $s \in S$ fix $y_{s,e}(n)$ to be equal to $y_{s,e}(n-1)$ for $e \leq n \cdot \delta$ for all selected event points representing resources used by users accepted in previous problem instances
- Fix user variables $x_{u,s,k}(n)$ to be equal to $x_{u,s,k}(n-1)$ for users who arrived before this iterations event points but are still active for any of the selected event points: $a(u) \leq n \cdot \delta, d(u) > n \cdot \delta$ for all $s \in S, k \in K(s)$ and all selected users as these represent users accepted or rejected in previous problem instances (i.e. users who have already been notified of acceptance)
- The users whose requests are accepted or rejected are: $u \in U_n$ where $n \cdot \delta \leq a(u) \leq (n+1)\delta$
- The profit obtained from newly confirmed requests is:

$$\sum_{\substack{u \in U \\ n \cdot \delta \leq a(u) \leq (n+1)\delta}} \left(p(u) \left(\sum_{s \in S} \sum_{k \in K(s)} x_{u,s,k}(n) \right) \right)$$

The total profit obtained after all iterations is defined as the profit from all users whose requests were confirmed in each iteration:

$$\sum_{n=0}^N \left(\sum_{\substack{u \in U \\ n \cdot \delta \leq a(u) \leq (n+1)\delta}} \left(p(u) \left(\sum_{s \in S} \sum_{k \in K(s)} x_{u,s,k}(n) \right) \right) \right) \quad (6.1)$$

6.1.2 Model Definition

This section states the integer program for the prebooking system using the integer program presented in §5.3. This model solves for all users currently in the system at any problem instance n . The solution of this model is only used to confirm the acceptance or rejection of users arriving within the next δ time, and determines on which servers those users will be placed.

In order to formulate the integer program for any instance n the time dependent sets are replaced by the appropriate problem instance subset. The user set U is replaced by U_n , the events set is replaced by E_n . The integer program from §5.3 is restated in the next sections with the problem instance notation for completeness.

6.1.2.1 Decision Variables

The model uses the following binary decision variables:

For each user $u \in U_n$, server $s \in S$, and every GPU core $k \in K(s)$ of server s , there is a binary variable $x_{u,s,k}$ indicating whether u is allocated to GPU core k on server s (from definition (5.1)):

$$x_{u,s,k} = \begin{cases} 1 & \text{if user } u \text{ is allocated to core } k \text{ of server } s \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

For each server $s \in S$, and event point $e \in E_n$, there is a integer variable $y_{s,e}$ indicating the number of web virtual machines being run on server s (from definition (5.3')):

$$y_{s,e} \geq 0 \quad (5.3')$$

Web users do not require GPU resources, and as such all variables for placing web users onto GPU cores are set to 0 (as in equation (5.19)):

$$x_{u,s,k} = 0 \quad \forall u \in U_n, s \in S, k \in K(s) : r(u) = \text{web}, k \neq 0 \quad (5.19^*)$$

All non-web users require a GPU core, and as such must be allocated to a GPU core when allocated to the server (as in equation (5.20)):

$$x_{u,s,0} = 0 \quad \forall u \in U_n, s \in S : r(u) \neq \text{web} \quad (5.20^*)$$

Users can not be placed on servers to which they cannot be assigned (as in equation (5.21)):

$$x_{u,s,k} = 0 \quad \forall u \in U_n, s \in S \setminus z(u), k \in K(s) \quad (5.21^*)$$

6.1.2.2 Integer Program

The objective is to maximise the total profit obtained by accepting users (mimicking the objective from (5.6')):

$$\max \sum_{u \in U_n} \left(p(u) \sum_{s \in S} \left(\sum_{k \in K(s)} x_{u,s,k} \right) \right) \quad (5.6'^*)$$

Subject to:

Each user $u \in U_n$ can only be allocated once (as in equation (5.7)):

$$\sum_{s \in S} \left(\sum_{k \in K(s)} x_{u,s,k} \right) \leq 1 \quad \forall u \in U_n \quad (5.7^*)$$

The internet cafe must have enough seats for all users at any point $e \in E_n$ (as in equation (5.8)):

$$\sum_{\substack{u \in U_n \\ a(u) \leq e < d(u)}} \sum_{s \in S} \sum_{k \in K(s)} x_{u,s,k} \leq \bar{N} \quad \forall e \in E_n \quad (5.8^*)$$

Enough virtual machines for the number of web users allocated to the server must be supplied (as in equation ((5.16)')):

$$\sum_{\substack{u \in U_n \\ a(u) \leq e < d(u)}} x_{u,s,0} \leq \bar{n}(s, \text{web}) y_{s,e} \quad \forall s \in S, e \in E_n \quad ((5.16)'^*)$$

Each server must have enough RAM and CPU to supply the virtual machines at any point $e \in E_n$ (as in equation (5.22)):

$$\forall s \in S, e \in E_n, q \in \{\text{cpu}, \text{ram}\} : \quad \sum_{\substack{u \in U_n \\ a(u) \leq e < d(u)}} \left(b(s, r(u), q) \sum_{k \in K(s): k \neq 0} x_{u,s,k} \right) + b(s, \text{web}, q) y_{s,e} \leq c(s, q) \quad (5.22^*)$$

Each server must have enough GPU resources to supply the virtual machines at any point $e \in E_n$ (as in equation (5.23)):

$$\sum_{\substack{u \in U_n \\ a(u) \leq e < d(u)}} (b(s, r(u), \text{gpu}) x_{u,s,k}) \leq 1 \quad \forall s \in S, k \in K(s), e \in E_n : k \neq 0 \quad (5.23^*)$$

6.2 Results

The prebooking model was solved for all problem instances in test set two as described in §4.3.5. The time-staged problem was solved using Gurobi 6.5.2 on a VM running Windows 10 which had sixteen 2.1 GHz CPU cores and 16 GB of RAM.

The test instances were all solved with a solve frequency of one hour on the basis that an internet cafe would want to send out confirmations of bookings as frequently as possible. The time horizon was varied from 1 hour up to 16 hours to quantify the extra profit gained from having more future information. Lastly, the total time Gurobi was given to solve was set at either 10 seconds or 120 seconds for each iteration n to get an idea of the solve time needed when deciding which requests to accept.

The batch solver is run with different settings on test data set two with each time horizon solve time combination run for 10 different realisations of the data, generated as described in §4.3.7. For this instance the number of iterations, $N = 24$ the total hours in test set twos problems.

6.2.1 Time Horizon

The first set of results compares the effect of changing the time horizon on solution quality compared to the optimal offline objective.

Table 6.1 shows the average objective achieved by the batch solver with 120 seconds solve time per iteration. This average objective is compared to the optimal offline objective. The average percentage of the offline objective obtained by all sizes and stays is 98.3% showing good potential for this solver with a prebooking system. Table 6.1 also has the averaged results for a time horizon of 2 hours. Even with this short time horizon, the batch solver can obtain over 95% of the offline objective for all test cases. The 2.5-hour average stay test cases all obtained around 99% of optimal with just a 2 hour time horizon.

As time horizon is increased, the problem gets larger, and eventually, the problem gets large enough that it has difficulty solving within the 120 second limit per iteration. At this point, the objectives start to worsen. The effect of the solve time limit becomes evident as the time horizon increases. While the integer program has more information and can make more informed decisions allowing it to come closer to the offline objective, it's increase in size makes it difficult to solve quickly. Eventually, the batch solver has enough information in

TABLE 6.1: Difference between optimal objective and batch solver results averaged over all tested time horizons and for the 2 hour time horizon over all test sets (10 problem instances for each number of users)

Users	Average Stay	Optimal	Prebooking Average	% diff +95% CI $\pm 0.21\%$	Time horizon 2 hours	% diff +95% CI $\pm 0.24\%$
300	2.5	1162.4	1155.5	99.4% $\pm 0.21\%$	1150.0	98.9% $\pm 0.24\%$
625	2.5	2429.1	2419.2	99.6% $\pm 0.52\%$	2411.4	99.3% $\pm 0.62\%$
1250	2.5	4895.9	4884.3	99.8% $\pm 0.18\%$	4873.8	99.5% $\pm 0.21\%$
1250	3.5	6068.4	5945.5	98.0% $\pm 0.53\%$	5969.3	98.4% $\pm 0.17\%$
1250	4.5	7184.7	6972.9	97.1% $\pm 0.55\%$	7035.8	97.9% $\pm 0.22\%$
1250	6.5	8754.5	8380.4	95.7% $\pm 0.82\%$	8475.6	96.8% $\pm 0.5\%$

each iteration that adding more information only acts to slow down the problem without improving the potential objective, creating worse solutions.

Figure 6.2 shows the percentage of the offline objective obtained by the batch solver for the realistic test set with 2.5 hour average stay times with total users of 300, 625, and 1250. There is a clear increase in the percentage of the offline objective obtained as the time horizon is increased for all test cases. These results plateau with a time horizon of 4 hours with minimal improvement beyond that value. It is worth noting that the 300 user test is significantly worse in terms of percentage of optimal, followed by the 625 user test. This is not because they are performing worse but rather because mistakes represent a bigger percentage of the objective for the 300 user cases - missing out on one user is 0.33% of all available users compared to 0.16% for 625 users, and 0.08% for 1250 users. This means the same number of mistakes alter the percentage difference more with less total users. The most notable feature is that all converge to over 99.5% after 4 hours.

Figure 6.3 shows the percentage of the offline objective obtained by the batch solver for the stress test with 1250 total users with average stay times of 2.5, 3.5, 4.5, and 6.5 hours. Notably the maximum percentage obtained by each is shifted towards a longer time horizon as stay time increases. The 2.5-hour average cases reach their peak percentage with a 4 hour

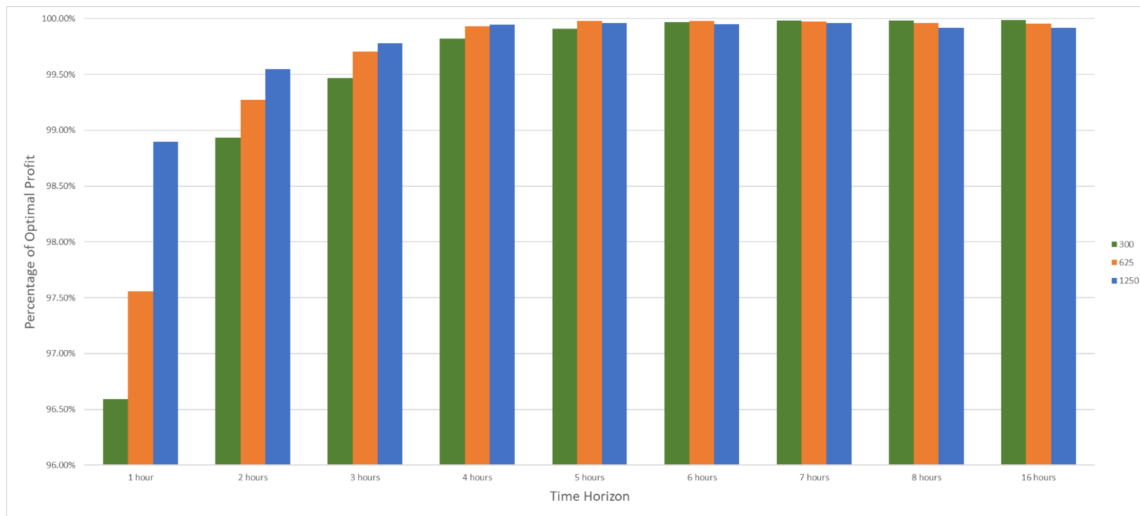


FIGURE 6.2: Average percentage difference of batch solver from optimal for the three different numbers of users

time horizon, with the 3.5-hour average cases reaching their peak after 6 hours, the 4.5 hour average cases after 7 hours, and the 6.5 hour average cases after 8 hours. The 3.5, 4.5 and 6.5-hour averages all experience a drop off in the percentage of the offline objective obtained as the time horizon increases as the problem space becomes too large to solve in 120 seconds.

One notable feature in figure 6.3 is the dip in the percentage obtained as time horizon increases before both the 4.5 and 6.5 average stay obtain their maximum percentage. The 4.5 hour average stay has a local minimum at 3 hours and the 6.5 hour average stay has one at 5 hours. This dip is because the problem size has reached a point where the algorithm is losing its ability to obtain near optimal solutions within the solve time while also not having enough information for the optimal solution to be close to the offline optimal. Once they have enough information from the longer time horizon, they can make a recovery in percentage obtained before being again overwhelmed by the problem size.

The maximum profit obtained from the batch solver appears to be related to the average time users stay. In figure 6.2 the maximum is reached at the same time horizon for all three internet cafe sizes. In figure 6.3 the maximum percentage of optimal profit is obtained with different time horizons. The maximum value can be found at the larger time horizons for the longer stay duration's. The 2.5 hour stay duration reaches its maximum with a 5 hour time

horizon, 3.5 hours with a 6 hour time horizon, 4.5 with a 7 hour time horizon, and 6.5 with a 8 hour time horizon.

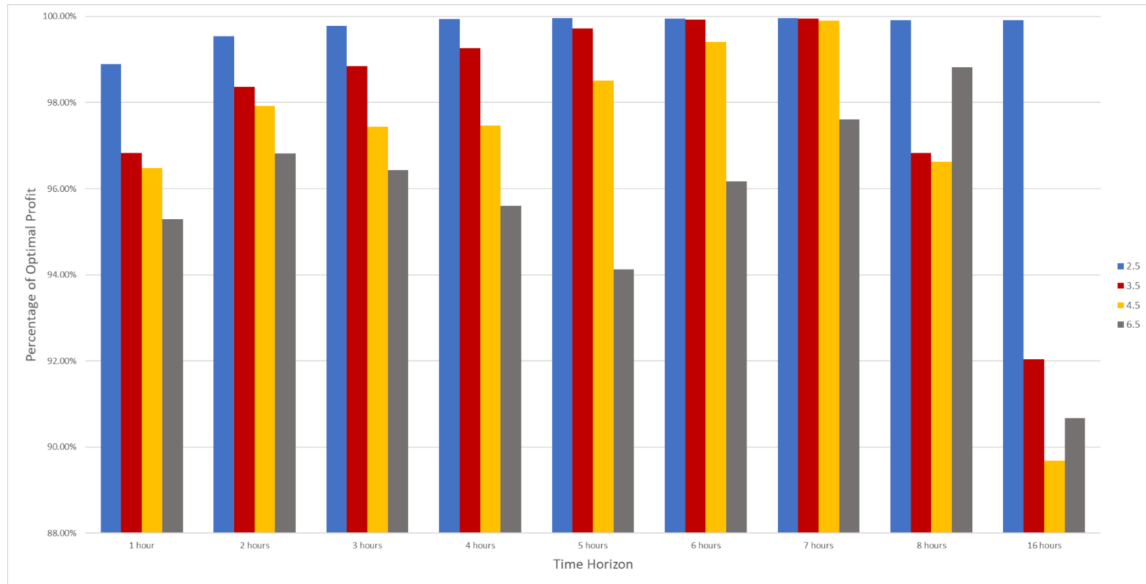


FIGURE 6.3: Average percentage difference of batch solver from optimal for the four different stay duration's

6.2.2 Solve Time

To see if the solver could be used aggressively to quickly place users the problem is tested with a 10 seconds solve time per iteration. This test was completed on all data sets in test set two.

Table 6.2 shows the average profit obtained with a 10 second solve time compared to both the offline and the batch solver with 120 seconds to solve. The 10 second solve times reduce the average percentage obtained by 3%. The 10 second solve time may be practical for a small 300 user internet cafe where the extra 110 seconds of solve only results in a 0.02% improvement and may also be practical for 625 users which only saw a 1% improvement. However, the large 1250 user internet cafe for low stay duration is 7.5% worse meaning there are significant gains from the extra solve time. The cases with a longer average stay performed over 10% worse for the 10 second solve than the 120 second solve such substantial differences make a 10 second solve time not worth considering.

TABLE 6.2: Comparison of 10 seconds and 120 second solve times for average objective from batch solver

Users	Average Stay	Optimal	10 sec solve	% diff +95% CI	120 sec solve	% diff +95% CI	Improvement
300	2.5	1162.4	1155.2	99.38% $\pm 0.21\%$	1155.5	99.40% $\pm 0.21\%$	0.02%
625	2.5	2429.1	2393.1	98.52% $\pm 0.63\%$	2419.2	99.59% $\pm 0.21\%$	1.07%
1250	2.5	4895.9	4518.4	92.29% $\pm 0.84\%$	4884.3	99.76% $\pm 0.18\%$	7.47%
1250	3.5	6068.4	5291.8	87.20% $\pm 1.09\%$	5945.5	97.97% $\pm 0.53\%$	10.77%
1250	4.5	7184.7	6101.7	84.93% $\pm 1.04\%$	6972.9	97.05% $\pm 0.55\%$	12.13%
1250	6.5	8754.5	7339.4	83.84% $\pm 1.02\%$	8380.4	95.73% $\pm 0.82\%$	11.89%

Figure 6.4 shows how the percentage of the offline optimal profit obtained by the batch solver changes as the time horizon is increased for different size internet cafes. The values from figure 6.2 are shown as faded bars versus result percentages for the 10 second solve. Initially, the percentages are very similar to the 10 second solve. However, as the time horizon increases the percentage begins to decrease with a more pronounced difference with more users. This difference is particularly significant in the 1250 user case where the difference gets as large as 16%.

6.2.3 Summary

This chapter has presented an algorithm which acts as an offline algorithm with limited information, creating a hybrid algorithm between online and offline. This algorithm would be used as part of a prebooking system for the cloud-based internet cafe.

A batch solver was developed which simulates users using the prebooking system, this was used to test the algorithm performance on test data set two as defined in §4.3.5. This simulation is completed using the methodology laid out in §6.1.1 for 10 realisations of each test case. The results give insight into how one may want to set the required time horizon and solve time based on internet cafe size and user stay duration. If the internet cafe is small

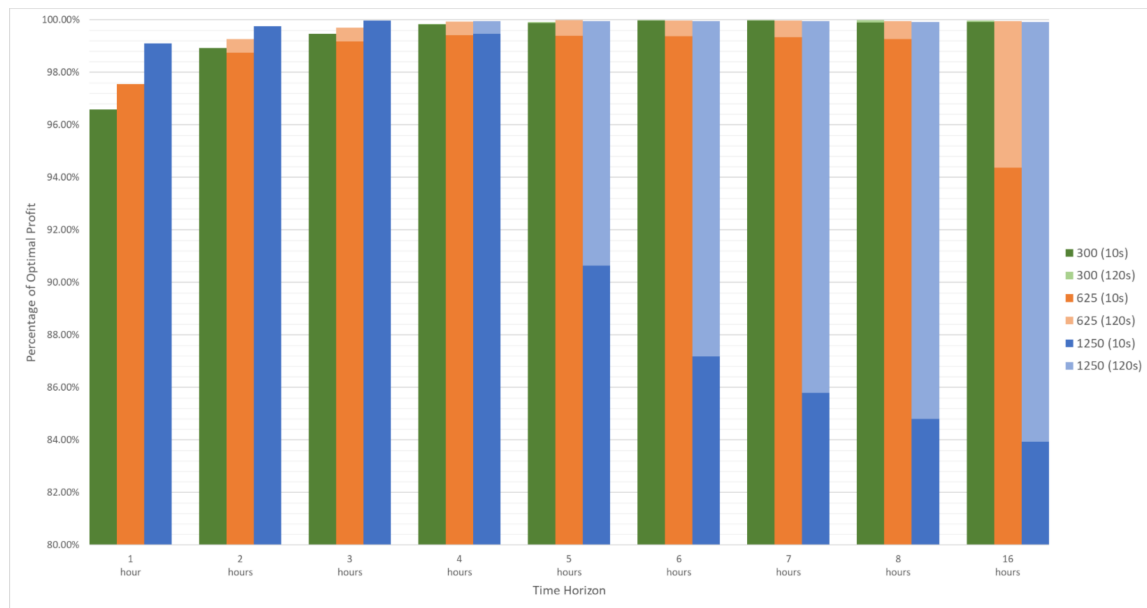


FIGURE 6.4: Average percentage difference of batch solver from optimal with 10 second vs 120 second solve time

or has a time horizon of 3 hours or less, it is possible to give solutions to the problem within 10 seconds.

For larger internet cafes ideally, a longer solve time would be given. The longer users stay on average in the internet cafe, the longer time horizon is desired and as such the longer users need to book in advance. It appears a good general rule is to set the time horizon one hour longer than the average duration users stay, however, this may vary depending on internet cafe size. The prebooking system has the potential to service internet cafes where there are consistent users who can book their arrivals in advance. However, if arrivals are unknown then another online strategy is needed to place these more random users.

Chapter 7

Online Algorithm

This chapter presents the online algorithms developed for placing users in real time for a cloud-based internet cafe. Greedy algorithms are presented which place users onto servers and only turn users away when no space is available on any server. Three strategies are shown for selecting the server to place a user. A second competitive algorithm is presented which uses an exponential weight function to decide which users to accept or reject and where to place them.

All algorithms are used to solve the online version of the cloud resource allocation problem described in chapter 4 on test data set two from §4.3.5. The results of these algorithms are presented alongside the results for the offline algorithm from chapter 5 to gauge the relative performance of the different algorithms. A summary of problem inputs is given in the notation section at the start of the thesis.

Users are allocated to seats in the cafe or turned away by these algorithms in the order they arrive at the internet cafe. The algorithms must decide if they accept or reject the user and if accepted where to place them before the next user is presented to the algorithm. These algorithms must use imperfect information to make decisions.

7.1 Greedy Algorithms

A greedy algorithm always attempts to place a user on a server and will not reject a user unless no resources are available.

Greedy algorithms pack users onto servers with available resources without considering future arrivals. The most straightforward algorithm is first fit where newly arrived users are placed onto the first server found with spare capacity. Alternative greedy algorithms (to first fit) search all servers to find the best matching server for a chosen condition. For this problem there are two sensible conditions: fill first, and fill last. With fill first, the users are placed on servers such that all resources on a server are utilised before placing users onto a new server. Fill last places users onto servers such that all servers increase in utilisation as evenly as possible.

In order to run any greedy algorithm we must first define the set of all time points within user u 's stay as $T(u) = \{t \in T : a(u) \leq t < d(u)\}$.

A first fit algorithms runs as follows: When a new user $u \in U$ arrives, search through the list of possible servers $z(u)$ for a server with spare capacity for the user (ordered by ascending server performance), if a server is found place the user onto that server. Define $i_{s,q,t}$, $s \in S, q \in Q, t \in T$, as the quantity of resource q being used on server s at time t . Then a user u can only be placed on a server $s \in z(u)$ if

$$i_{s,q,t} + b(s, q, u, t) \leq c(s, q) \quad \forall s \in z(u), q \in Q, t \in T(u)$$

This equation ensures that the resources currently being used by the server ($i_{s,q,t}$), and the resources which the new user will consume ($b(s, q, u, t)$) are less than or equal to the available resources ($c(s, q)$).

Once a server s' is found which meets these conditions update its resource usage $i_{s',q,t}$ to match the new utilisation:

$$i_{s',q,t} = i_{s',q,t} + b(s', q, u, t) \quad \forall q \in Q, t \in T(u)$$

The two other sensible conditions can be defined as follows:

1) Fill first: where new users are placed on the server with the least capacity to spare. Select the server with the highest average percentage utilisation (with the new user added) over all resources for the time period the user stays (given by event points $(T(u))$ over the users duration $(j(u))$):

$$\max_{s \in z(u)} \frac{1}{|Q|j(u)} \sum_{q \in Q} \sum_{t \in T(u)} \left(\frac{i_{s,q,t} + b(s, q, u, t)}{c(s, q)} \right)$$

2) Fill last: where new users are placed on servers with the smallest quantity of resources currently in use. Select the server with the lowest average percentage utilisation (with the new user added) over all resources for the time period the user stays:

$$\min_{s \in z(u)} \frac{1}{|Q|j(u)} \sum_{q \in Q} \sum_{t \in T(u)} \left(\frac{i_{s,q,t} + b(s, q, u, t)}{c(s, q)} \right)$$

An advantage of fill first is that it keeps servers idle when the internet cafe is not at capacity, potentially saving power costs for the owners. Conversely, fill last could offer users improved performance when the internet cafe is not at capacity.

7.2 Competitive Algorithm

A greedy algorithm may not always perform well given certain conditions. For example, suppose at 3pm students arrive at the internet cafe and stay until 6pm, when they leave for dinner. Suppose also that at around 5pm workers arrive as they finish work for the day, and they wish to stay until midnight. If the greedy algorithm has allowed the internet cafe to be

filled with students then there may be no capacity for these workers. In the same manner, all the students cannot be rejected because it is not guaranteed any workers show up at 5pm. There is a requirement for an algorithm that can strike a balance between the users present now and the users that may arrive in the future.

The following algorithm is based on other online competitive (or adversarial) algorithms which guarantee a minimum percentage of the optimal objective. The assumptions and strategies required, as well as proofs for competitiveness, are adaptations of the phone call routing online optimisation problem [115]. The competitive algorithm uses an exponential cost function. Initially, all users are accepted but as resources on servers near capacity less profitable users are no longer accepted with the profitability required for acceptance rising as spare capacity diminishes. In this way, spare capacity is reserved for potential high-profit users while some lower profit users are still accepted.

This algorithm requires assumptions concerning the complexities of the GPU virtualisation. This algorithm considers the server GPU to be a single continuous resource, rather than a group of individual cores. Nevertheless, it still yields feasible solutions.

In this problem, web users may share a single virtual machine. The first user consumes resources when the virtual machine is started. Sequential users consume no additional resources as they are added to the shared virtual machine until it is at capacity. As previously explained in §4.1 it is desirable to have these shared virtual machines because these users individually only require a small quantity of resources while active. However, when opening new applications the resources needed are significantly higher than at other moments. By creating shared virtual machines rather than individual virtual machines, the virtual machine can supply the “spike” usage while also efficiently using resources during standard usage.

Unfortunately, the competitive algorithm requires that resources supplied are continuous and that each user is independent of all others. Although users are not independent on shared virtual machines, it can be shown that these users only cause the algorithm to be worse by at

most a factor equal to the number of users that can share these virtual machines. The effect of these users is discussed in more detail in §7.2.3.

7.2.1 Algorithm

This section describes the competitive algorithm and the bounds which must be placed onto the algorithm in order to prove the feasibility and competitive ratio of the algorithm.

The algorithm is designed such that it will accept user u onto a server $s \in z(u)$ when:

$$\sum_{q \in Q} \sum_{t \in T} \frac{b(s, q, u, t)}{c(s, q)} V_{s, q, u, t} \leq p(u) \quad (7.1)$$

where $V_{s, q, u, t}$ is an exponential weight function determined from the resources currently being used. This acceptance equation (7.1) exponentially weights the resources required by the new user u and the resources currently used through V against the profit that u obtains. Using this acceptance rule the profit the algorithm obtains will be at least

$$\frac{1}{\log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)}$$

of the profit gained by the offline algorithm where:

μ represents the maximum profit that could be gained by a single user; and

λ is a lower bound on the profit per resource used.

In order to implement and prove the performance of the algorithm, the exponential weight function V used in the algorithm needs to be defined. First it is necessary to bound the profit and resources used to a limited quantity. This process for defining V and μ (as well as λ and F) is described next.

Define λ and F First, the average profit (from a user) per unit of resources used per time unit is bounded below by λ and above by F for all users, servers, (utilised) resources and

time points:

$$0 < \lambda \leq \frac{1}{|Q|} \frac{p(u)}{b(s, q, u, t)j(u)} \leq F \quad \forall u \in U, s \in z(u), q \in Q, t \in T : b(s, q, u, t) \neq 0 \quad (7.2)$$

The value F and λ represent the maximum and minimum possible profitability of users per unit of resource consumed per time respectively. In an internet cafe the ratio of profit to bandwidth for different services is similar, giving minimal variance between users in the profit assumption, thus helping to limit the value of F .

Defining μ from F In (7.3) the value of μ is determined from F :

$$\mu = 2(JF|Q| + 1) \quad (7.3)$$

This defines μ as a factor of the maximum time a user stays J , the number of different resources they consume $|Q|$, and maximum profitability per resource consumed F . This is defined to simplify the proofs for feasibility and competitiveness.

Limiting user demand using μ As servers have large quantities of resources to be supplied to many users, each user only consumes a small quantity of the total resources on a server. For this reason it is acceptable to assume that the quantity of each resource used must not be over a fixed percentage of the total available capacity to allow sufficient flexibility when allocating users. This percentage is given by a factor of μ , the maximum profit that can be gained from a single user:

$$b(s, q, u, t) \leq \frac{c(s, q)}{\log_2 \mu} \quad \forall u \in U, s \in z(u), q \in Q, t \in T \quad (7.4)$$

This bandwidth assumption holds up in a real world cloud-based internet cafe where the motivation is that multiple users will be placed on each server with no one user taking up a large percentage of available resources which would reduce flexibility.

Defining V from μ The algorithm uses μ alongside system variables to allocate new users using a normalised load function where the resources utilised is given as a fraction of total resources. Let s be the server to which a given user u is allocated ($= 0$ if u is rejected) and define $s(u')$ as the server allocated to any user u' that arrived before u . Then the $L_{s,q,u,t}$ is the normalised load (the proportion of total capacity being used) before user u is accepted on server s is given by

$$L_{s,q,u,t} = \sum_{\substack{a(u') < a(u) \\ s(u') = s}} \frac{b(s, q, u', t)}{c(s, q)}$$

Using this normalised load an exponential edge cost equation that defines the exponential weight function V can be specified. The exponential weight function makes user acceptance less likely as more users are allocated to the servers:

$$V_{s,q,u,t} = c(s, q)(\mu^{L_{s,q,u,t}} - 1) \quad (7.5)$$

This exponential weight function then determines whether a user u is accepted to be allocated to a server $s \in z(u)$ as described earlier in (7.1):

$$\sum_{q \in Q} \sum_{t \in T} \frac{b(s, q, u, t)}{c(s, q)} V_{s,q,u,t} \leq p(u) \quad (7.1)$$

If accepted then assign the user to such a server.

7.2.2 Proofs

To utilise this algorithm, it is necessary to prove that it will output a feasible solution, see Proposition 7.1. To show the algorithm provides effective solutions it is necessary to prove the competitive ratio of $\frac{1}{\log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)}$ when compared to the offline objective, see Theorem 7.4.

Proposition 7.1. *The algorithm will always output a feasible solution.*

Proof. Assume that u is assigned to server s and is the first user that would cause a violation of resource q on that server, i.e., this assignment causes an infeasible solution. Hence, the relative load before accepting u , $L_{s,q,u,t}$, plus the relative load that u adds, $\frac{b(s,q,u,t)}{c(s,q)}$, must exceed one at some point during their stay, so:

$$L_{s,q,u,t} > 1 - \frac{b(s,q,u,t)}{c(s,q)} \text{ for some } a(u) \leq t \leq d(u) \quad (7.6)$$

Consider a slightly rearranged version of the exponential cost equation (7.5) that is used to decide upon accepting users to servers:

$$\frac{V_{s,q,u,t}}{c(s,q)} = \mu^{L_{s,q,u,t}} - 1 \quad (7.5')$$

Note that $\mu > 1$ (from (7.3) and the fact that $J, F, |Q| \geq 0$), so (7.5') and (7.6) combine to give:

$$\frac{V_{s,q,u,t}}{c(s,q)} > \mu^{1 - \frac{b(s,q,u,t)}{c(s,q)}} - 1 \quad (7.7)$$

Incorporating the bandwidth assumption from (7.4):

$$\mu^{1 - \frac{b(s,q,u,t)}{c(s,q)}} - 1 \geq \mu^{1 - \frac{1}{\log_2 \mu}} - 1 = \frac{\mu}{2} - 1 \quad (7.8)$$

From the definition of μ in (7.3):

$$\frac{\mu}{2} - 1 = JF|Q| \quad (7.9)$$

Combining (7.7)-(7.9) and multiplying the entire equation by $b(s,q,u,t)$ gives:

$$\frac{b(s,q,u,t)}{c(s,q)} V_{s,q,u,t} > JF|Q|b(s,q,u,t)$$

Finally, under profit assumption (7.2) and $J \geq j(u)$ gives:

$$\frac{b(s, q, u, t)}{c(s, q)} V_{s, q, u, t} > JF|Q|b(s, q, u, t) \geq j(u)F|Q|b(s, q, u, t) \geq p(u)$$

Hence u could not have been assigned to server s by the algorithm as the acceptance equation (7.1) is violated. This provides a contradiction, meaning the algorithm will always produce feasible solution. \square

Lemmas 7.2 and 7.3 that follow are preliminaries for Theorem 1 that proves the competitive ratio for the algorithm.

Lemma 7.2. *The sum of all exponential costs provides a lower bound for the total profit.*

Let A be a set of users already accepted by the algorithm, with α being the most recent user accepted or rejected. Then the profit gained from accepting A , $\sum_{u \in A} p(u)$, is bounded as follows:

$$\left(1 + \frac{1}{\lambda}\right) \log_2 \mu \sum_{u \in A} p(u) \geq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s, q, \alpha+1, t} \quad (7.10)$$

Proof. By induction on α :

For $\alpha = 0$ (i.e., no users considered yet): both sides resolve to 0 as without users the set A is empty and the servers have no load.

At step α : assuming (7.10) holds for $\alpha - 1$ then:

1. if user α is rejected: the left-hand side of (7.10) is unchanged as A is unchanged and the right-hand side of (7.10) is unchanged as the load on all the servers $s \in S$ is unchanged, so (7.10) holds for α ;
2. if user α is accepted: the left-hand side of (7.10) increases by $(1 + \frac{1}{\lambda}) \log_2 \mu \cdot p(\alpha)$ and the right-hand side increases by

$\sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t})$, so (7.10) holds for α if the left-hand side change is \geq the right-hand side change.

Therefore, it is sufficient to show that:

$$(1 + \frac{1}{\lambda}) \log_2 \mu \cdot p(\alpha) \geq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t}) \quad (7.11)$$

Consider the change in (7.5) for resource q at time t on the server s' where α is assigned:

$$\begin{aligned} V_{s',q,\alpha+1,t} - V_{s',q,\alpha,t} &= c(s', q) \left(\mu^{L_{s',q,\alpha,t} + \frac{b(s',q,\alpha,t)}{c(s',q)}} - \mu^{L_{s',q,\alpha,t}} \right) \\ &= c(s', q) \mu^{L_{s',q,\alpha,t}} \left(\mu^{\frac{b(s',q,\alpha,t)}{c(s',q)}} - 1 \right) \\ &= c(s', q) \mu^{L_{s',q,\alpha,t}} \left(2^{(\log_2 \mu) \frac{b(s',q,\alpha,t)}{c(s',q)}} - 1 \right) \end{aligned}$$

From the bandwidth assumption (7.4) we know that $b(s', q, \alpha, t) \leq \frac{c(s', q)}{\log_2 \mu}$, i.e., $(\log_2 \mu) \frac{b(s', q, \alpha, t)}{c(s', q)} \leq 1$. Since $2^x - 1 \leq x$ for $0 \leq x \leq 1$, we get:

$$\begin{aligned} V_{s',q,\alpha+1,t} - V_{s',q,\alpha,t} &\leq c(s', q) \mu^{L_{s',q,\alpha,t}} (\log_2 \mu) \frac{b(s', q, \alpha, t)}{c(s', q)} \\ &\leq (\log_2 \mu) \mu^{L_{s',q,\alpha,t}} b(s', q, \alpha, t) \end{aligned}$$

Rearranging the exponential cost equation (7.5) to get $\mu^{L_{s',q,\alpha,t}} = \frac{V_{s',q,\alpha,t}}{c(s', q)} + 1$ and substituting gives:

$$V_{s',q,\alpha+1,t} - V_{s',q,\alpha,t} \leq (\log_2 \mu) \left(V_{s',q,\alpha,t} \frac{b(s', q, \alpha, t)}{c(s', q)} + b(s', q, \alpha, t) \right) \quad (7.12)$$

Combining (7.12) with the fact that the value for change is 0 in the exponential equations or load for any $s \neq s'$ we can sum over $t \in T$ and $q \in Q$ to obtain:

$$\begin{aligned} \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t}) &\leq \log_2 \mu \sum_{t \in T} \sum_{q \in Q} \left(V_{s',q,\alpha,t} \frac{b(s',q,\alpha,t)}{c(s',q)} + b(s',q,\alpha,t) \right) \\ &\leq \log_2 \mu \left(\sum_{t \in T} \sum_{q \in Q} V_{s',q,\alpha,t} \frac{b(s',q,\alpha,t)}{c(s',q)} + \sum_{t \in T} \sum_{q \in Q} b(s',q,\alpha,t) \right) \end{aligned} \quad (7.13)$$

The first term within the parentheses of the right-hand side of (7.13) can be simplified using the fact that α was accepted on s' and the acceptance equation (7.1). The second term can be simplified by rearranging the lower bound from the profit assumption (7.2): $b(s',q,\alpha,t) \leq \frac{1}{\lambda|Q|} \frac{p(\alpha)}{j(\alpha)}$ which only applies for $b(s',q,\alpha,t) \neq 0$. This gives:

$$\begin{aligned} \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t}) &\leq \log_2 \mu \left(\sum_{t \in T} \sum_{q \in Q} V_{s',q,\alpha,t} \frac{b(s',q,\alpha,t)}{c(s',q)} + \sum_{t \in T} \sum_{q \in Q} b(s',q,\alpha,t) \right) \\ &\leq \log_2 \mu \left(p(\alpha) + \sum_{\substack{t \in T \\ b(s',q,\alpha,t) \neq 0}} \sum_{q \in Q} \frac{1}{\lambda|Q|} \frac{p(\alpha)}{j(\alpha)} \right) \\ &\leq \log_2 \mu \left(p(\alpha) + \frac{1}{\lambda} \frac{p(\alpha)}{j(\alpha)} \sum_{\substack{t \in T \\ b(s',q,\alpha,t) \neq 0}} 1 \right) \end{aligned} \quad (7.14)$$

$\sum_{t \in T: b(s',q,\alpha,t) \neq 0} 1 = \sum_{t=a(\alpha)}^{d(\alpha)} 1 = j(\alpha)$ as the bandwidth is only non-zero when the user is active. This gives:

$$\log_2 \mu \left(p(\alpha) + \frac{1}{\lambda} \frac{p(\alpha)}{j(\alpha)} \sum_{\substack{t \in T \\ b(s',q,\alpha,t) \neq 0}} 1 \right) = \log_2 (\mu) (p(\alpha) + \frac{1}{\lambda} p(\alpha)) \quad (7.15)$$

Combining (7.14) and (7.15):

$$\sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t}) \leq \log_2(\mu)(p(\alpha) + \frac{1}{\lambda}p(\alpha))$$

$$\sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} (V_{s,q,\alpha+1,t} - V_{s,q,\alpha,t}) \leq (1 + \frac{1}{\lambda})p(\alpha) \log_2(\mu)$$

Matching (7.11) proving by induction Lemma 7.2 (7.10) □

Lemma 7.3. *The sum of all exponential costs provides an upper bound for the optimal offline profit. Let H be the set of users accepted by the offline algorithm but not the online algorithm. Let β be the most recent user added to H set then:*

$$\sum_{u \in H} p(u) \leq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,\beta,t} \quad (7.16)$$

Proof. Let $s'(u)$ be the server used by the offline algorithm to host a user $u \in H$. The fact that each u was not accepted by the online algorithm implies that the acceptance equation (7.1) was false for each $u \in H$:

$$p(u) < \sum_{q \in Q} \sum_{t \in T} \frac{b(s'(u), q, u, t)}{c(s'(u), q)} V_{s'(u),q,u,t}, \quad u \in H \quad (7.17)$$

As V is monotonically increasing with respect to H and β is the last member added to H :

$$p(u) < \sum_{q \in Q} \sum_{t \in T} \frac{b(s'(u), q, u, t)}{c(s'(u), q)} V_{s'(u),q,\beta,t}, \quad u \in H \quad (7.18)$$

Using (7.18) and summing over all $u \in H$:

$$\sum_{u \in H} p(u) < \sum_{u \in H} \sum_{q \in Q} \sum_{t \in T} \frac{b(s'(u), q, u, t)}{c(s'(u), q)} V_{s'(u),q,\beta,t}$$

Consider summing over $s \in S$ and using an indicator function to identify $s'(u)$:

$$\mathbb{1}_A = \begin{cases} 1 & A \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_{u \in H} p(u) < \sum_{u \in H} \sum_{q \in Q} \sum_{t \in T} \sum_{s \in S} \mathbb{1}_{s'(u)=s} \left(\frac{b(s, q, u, t)}{c(s, q)} V_{s, q, \beta, t} \right)$$

The order of the sums can be swapped as there are no conditions of the set being summed over:

$$\sum_{u \in H} p(u) < \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} \sum_{u \in H} \mathbb{1}_{s'(u)=s} \left(\frac{b(s, q, u, t)}{c(s, q)} V_{s, q, \beta, t} \right)$$

The exponential cost does not refer to u , i.e., it is constant within the sum over $u \in H$, so it can be moved out of that sum term:

$$\sum_{u \in H} p(u) < \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s, q, \beta, t} \sum_{u \in H} \mathbb{1}_{s'(u)=s} \frac{b(s, q, u, t)}{c(s, q)}$$

Finally, consider the following term for any given $s \in S$:

$$\begin{aligned} \sum_{u \in H} \mathbb{1}_{s'(u)=s} \frac{b(s, q, u, t)}{c(s, q)} &= \sum_{\substack{u \in H \\ s'(u)=s}} \frac{b(s, q, u, t)}{c(s, q)} = \frac{1}{c(s, q)} \sum_{\substack{u \in H \\ s'(u)=s}} b(s, q, u, t) \\ &= \frac{\text{total used resource for } q \text{ on } s \text{ from users placed on } s \text{ in } H}{\text{capacity of } q \text{ on } s} \\ &\leq 1 \text{ (as the offline algorithm would not overload a server).} \end{aligned}$$

Therefore:

$$\sum_{u \in H} p(u) < \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s, q, \beta, t} \sum_{u \in H} \mathbb{1}_{s'(u)=s} \frac{b(s, q, u, t)}{c(s, q)} \leq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s, q, \beta, t}$$

which proves Lemma 7.3. □

Theorem 7.4. *The algorithm obtains at least a $\log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)$ fraction of the profit of the offline algorithm.*

Proof. The profit of the offline algorithm (OPT) can be bounded by:

$$\begin{aligned}
\text{OPT} &= \text{profit from users accepted by offline algorithm} \\
&= \text{profit from users accepted by offline, not online algorithm} \\
&\quad + \text{profit from users accepted by both offline and online algorithms} \\
&\leq \underbrace{\sum_{u \in H} p(u)}_{\text{accepted by offline, not online}} + \underbrace{\sum_{u \in A} p(u)}_{\text{accepted by online}} \\
&\quad \text{(note that } p(u) \text{ of accepted users is } \geq 0 \text{ or they would be rejected)}
\end{aligned}$$

Using Lemma 7.3, (7.16), the first term on the right-hand side is bounded by profit bound can be upper bounded by

$\sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,\beta,t}$ where β is the most recent user added to H , so:

$$\text{OPT} \leq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,\beta,t} + \sum_{u \in A} p(u)$$

Given the exponential cost value $V_{s,q,U+1,t}$ when all users have been considered, i.e., the final cost of resource q on server s , we know that the monotonicity of V gives that $\forall s \in S, q \in Q, t \in T : V_{s,q,U+1,t} \geq V_{s,q,\beta,t}$, so:

$$\text{OPT} \leq \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,U+1,t} + \sum_{u \in A} p(u)$$

Lemma 7.2, (7.10), combined with the fact that rejected users don't alter the exponential cost values gives:

$$\begin{aligned} \text{OPT} &\leq \underbrace{\sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,U+1,t}} + \sum_{u \in A} p(u) \leq (1 + \frac{1}{\lambda}) \log_2 \mu \sum_{u \in A} p(u) + \sum_{u \in A} p(u) \\ &= \sum_{t \in T} \sum_{s \in S} \sum_{q \in Q} V_{s,q,\alpha+1,t} \end{aligned}$$

where α is the last user accepted by the online algorithm

$$\begin{aligned} \text{OPT} &\leq (1 + \frac{1}{\lambda}) \log_2 \mu \sum_{u \in A} p(u) + \sum_{u \in A} p(u) = ((1 + \frac{1}{\lambda}) \log_2 \mu + 1) \sum_{u \in A} p(u) \\ &\leq ((1 + \frac{1}{\lambda}) \log_2 \mu + \log_2 2) \sum_{u \in A} p(u) \\ &\leq \log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right) \sum_{u \in A} p(u) \end{aligned}$$

Proving the competitive ratio for this problem □

7.2.3 Grouped Users

If we have some users which can be grouped on a single virtual machine, then it is difficult to determine how to value these users. In fact, the exponential algorithm has no way to determine when to accept these users. A simple fix is to value the users at $\bar{n} \cdot \pi(u)$ where \bar{n} is the number in each group. Then treat each of these users as if they use all the resources on the shared VM (i.e., no sharing of resources).

From this altered profit, value define $\text{ALG}(U)$ and $\text{OPT}(U)$ as the real profit obtained by the algorithm and offline optimum respectively and $\text{ALG}'(U)$ the altered profit the algorithm obtains.

Theorem 7.5. *The algorithm with modified profit values obtains at least a $\frac{1}{\bar{n} \log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)}$ fraction of the profit of the offline algorithm.*

Proof. By multiplying the profits of these users by \bar{n} the normal algorithm will always obtain profit at least as large as the algorithm with altered profits scaled back down by \bar{n} :

$$\text{ALG}(U) \geq \frac{\text{ALG}'(U)}{\bar{n}}$$

From the competitive ratio:

$$\text{OPT}(U) \geq \frac{1}{\log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)} \text{ALG}(U)$$

$$\text{OPT}(U) \geq \frac{1}{\bar{n} \cdot \log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)} \text{ALG}'(U)$$

Therefore this modified algorithm changes the competitive ratio by a factor of \bar{n} . \square

7.2.4 Aggressive Improvement

Unfortunately, the competitive algorithm is often too conservative when accepting users. The algorithm will often choose to reject a user of medium value once the server is only partially filled. To improve the performance of the algorithm, it is necessary to make it more aggressively accept users. This aggression can be achieved, firstly by allowing the grouped users to again share resources and, secondly, by altering the acceptance condition to encourage accepting less profitable users.

While treating group users as larger single units provides a guaranteed competitive ratio, it wastes resources by over-allocating resources per user. Instead, an aggressive slant can be taken where the first grouped user is accepted using the competitive strategy, and additional users are greedily added to the shared virtual machine until it is at capacity. At which point opening a new virtual machine is considered once again using the competitive strategy.

In addition, the acceptance condition can be relaxed in order to consider less profitable users. The relaxed acceptance condition can be implemented by scaling the profit in the acceptance equation (7.1) by a factor (κ) to encourage the acceptance of a greater number of users. This results in an adjusted acceptance equation:

$$\sum_{q \in Q} \sum_{t \in T} \frac{b(s, q, u, t)}{c(s, q)} V_{s, q, u, t} \leq \kappa p(u)$$

This aggressive problem still makes use of the exponential costs to consider the acceptance or rejection of users but no longer offers a guaranteed competitive ratio. In fact, the algorithm no longer guarantees feasibility and it is important to check that no resources are overloaded when allocating a new user to a server. In exchange for this deterioration in “worst case” performance the algorithm can improve its objective for real-world test cases.

7.3 Results

These algorithms were tested on the data sets presented in §4.3.5. In order to test the variance of the algorithm performance 10 random realisations of each test case in test set two were generated as described in §4.3.7. The profits obtained by these algorithms are compared to the profits of the optimal offline algorithm presented in §5.5.2.

To create solutions for this section: 1) all test case realisations were solved using the greedy and competitive algorithm, 2) results were calculated as a percentage of the offline optimal, and 3) profits and percentages were averaged within each test case including 95% confidence intervals. These average profit and percentage of offline optimal values are then used to compare algorithm performance in this section.

Both algorithms were implemented using C++ with no special libraries required. All algorithms can allocate new users instantaneously and only take a few seconds to solve for the full 24 hour period for all test cases. These algorithms have the added advantage of being

practical for usage in a cloud-based internet cafe with the ability to place users instantly and running on free software.

Table 7.1 contains the competitive ratio and related coefficients and constants for test set two. Coefficients λ and F represent the minimum and maximum profit per unit of resource consumed. Constant $|Q|$ is the number of resource types in the problem and J is the maximum duration any one user stays. Finally, μ represents the maximum percentage of resource capacity any one user consumes. Overall these factors are combined using the equations in table 7.1 to give the competitive ratio or the minimum percentage of the optimal offline objective that can be obtained by the algorithm.

The competitive ratios for all test cases are below 0.1% due the problem complexity, however, results show good performance from the competitive algorithm. The low competitive ratio does cause the algorithm to behave conservatively when accepting users and as such the aggressive constant, κ was set to 600 for the aggressive competitive algorithm for all results presented.

TABLE 7.1: Competitive ratio and calculation components for different user stay durations. Symbols, λ , F , $|Q|$, J , and μ are defined in §7.2.1

Coefficient	λ	F	$ Q $	J	μ	Competitive Ratio
Equation	$\min \frac{1}{ Q } \frac{p(u)}{b(s,q,u,t)j(u)}$	$\max \frac{1}{ Q } \frac{p(u)}{b(s,q,u,t)j(u)}$			$2(JF Q + 1)$	$\frac{1}{\log_2 \left(2\mu^{1+\frac{1}{\lambda}} \right)}$
2.5 hour stay	0.0095	0.304	3	8	16.6	0.06%
3.5 hour stay	0.0075	0.24	3	10	16.4	0.05%
4.5 hour stay	0.0065	0.208	3	12	17.0	0.04%
6.5 hour stay	0.0055	0.176	3	14	16.8	0.03%

7.3.1 Internet Cafe Size

The first set of test cases considered are the realistic test set with different size internet cafes and a constant gamma distribution determining stay duration as defined in §4.3.6.2 with an

average of 2.5 hours. Three internet cafe sizes are considered: 120, 250, and 500 seat internet cafes.

Tables 7.2 and 7.3 show the total profit received from the optimal algorithm, and the value plus percentage differences from optimal for the greedy algorithms and competitive online algorithms respectively. Since the number of servers has been selected to be capable of fitting as many users as there are seats, all algorithms can perform relatively competitively for this real-world data set. This result is reflected by the fact that the worst result is 85.4% of the optimal. However, due to the large number of users in internet cafes, small percentage differences can result in significant differences in profits over time. It is interesting to note the percentage of profit grows as the number of seats in the internet cafe increases due to an increased number of servers making the solutions less vulnerable to accepting users which then forces the algorithm to reject users which the offline optimal accepted.

All greedy algorithms perform similarly. Interestingly the first fit works slightly better than fill first or fill last on average. Both fill first and fill last have better profits than first fit in some of the individual realisations, but on average have around 1% worse performance. This difference may be explained by the ordering of servers in first fit by performance, causing the low performance servers to fill first leaving space for medium and high users on the high performance servers.

The standard competitive algorithm performs significantly worse than the greedy algorithms, only receiving 87% of the available profit compared to 96% for each greedy algorithm. When the aggressive adjustment is made to the competitive algorithm, it performs significantly better obtaining 95% of the optimal profit, and while it still performs worse than the greedy algorithms on average, it does outperform the greedy algorithms in multiple realisations.

For all sizes of internet cafe the greedy algorithms obtain a large percentage of the offline optimal profit, more than both the competitive and aggressive competitive algorithms. The aggressive competitive algorithm is able to close the gap and outperforms the competitive algorithm which is too conservative with its acceptance. In the next section the effect of

TABLE 7.2: Greedy algorithm performance for different size internet cafes, average stay = 2.5 hours with 95% confidence intervals

Objective	Optimal	First Fit		Fill First		Fill Last	
120 Seats	1162.4	1117.2	96.1% ±0.74%	1111.2	95.6% ±1.1%	1115.1	95.9% ±1.25%
250 Seats	2429.1	2355.2	97.0% ±0.76%	2333.8	96.1% ±0.89%	2354.1	96.9% ±0.8%
500 Seats	4895.9	4769.5	97.4% ±0.62%	4727.4	96.6% ±0.69%	4764.2	97.3% ±0.53%
Average			96.8% ±0.7%		96.1% ±0.9%		96.7% ±0.86%

TABLE 7.3: Competitive algorithm performance for different size internet cafes, average stay = 2.5 hours with 95% confidence intervals

Objective	Optimal	Competitive		Aggressive Competitive	
120 Seats	1162.4	992.2	85.4% ±2.09%	1095.1	94.2% ±1.02%
250 Seats	2429.1	2114.0	87.0% ±1.48%	2330.3	95.9% ±0.63%
500 Seats	4895.9	4291.3	87.6% ±0.96%	4754.8	97.1% ±0.45%
Average			86.7% ±1.51%		95.8% ±0.7%

increasing user density in the internet cafe is tested by increasing the duration each user stays on average. This increases the value of each choice as the internet cafe is at capacity more frequently.

7.3.2 Stay Duration

The second group of test cases is the stress test considering an internet cafe with 500 seats for four different user stay duration distributions with 2.5, 3.5, 4.5, and 6.5 hour average stays. For each of these stay durations 10 random realisations are generated as described in §4.3.7.

Tables 7.4 and 7.5 show the objective values for the 500 seat internet cafe with increasing durations for users for both the greedy and competitive algorithms respectively. The result of the increasing average stay duration is that there are more users in the internet cafe at all time periods, often exceeding the available seats, and the potential for users to generate very high profit (for the internet cafe) from their high stay duration.

As users stay for longer all online algorithms obtain less of the total offline profit, falling from around 97% to 90% for the longest stay. When internet cafes are full it is much more important to accept the correct users and accepting any user as in the greedy algorithms can result in significantly different user acceptance choices when compared to the offline optimal.

TABLE 7.4: Greedy algorithm performance for different average stay durations, internet cafe size = 500 with 95% confidence intervals

Average Stay	Optimal	First Fit		Fill First		Fill Last	
2.5 Hour	4895.9	4769.5	97.4% $\pm 0.62\%$	4727.4	96.6% $\pm 0.69\%$	4764.2	97.3% $\pm 0.53\%$
3.5 Hour	6011.6	5665.6	94.2% $\pm 0.75\%$	5649.1	94.0% $\pm 0.66\%$	5674.4	94.4% $\pm 0.59\%$
4.5 Hour	7149.4	6611.8	92.5% $\pm 0.83\%$	6628.5	92.7% $\pm 0.52\%$	6608.3	92.4% $\pm 0.88\%$
6.5 Hour	8696.0	7816.8	89.9% $\pm 0.61\%$	7794.7	89.6% $\pm 0.63\%$	7825.2	90.0% $\pm 0.63\%$
Average			93.5% $\pm 0.7\%$		93.2% $\pm 0.63\%$		93.5% $\pm 0.66\%$

As the stay duration increases the aggressive competitive algorithm starts to outperform the greedy algorithm. With 6.5 hour average stay the aggressive competitive algorithm ends up with 0.5% extra profit over the greedy algorithm, a statistically significant difference ($p\text{-value} < 0.001$). The extra profit shows the value of selectively rejecting some users. For this reason, the competitive algorithm is superior when resources are highly contested throughout the day.

TABLE 7.5: Competitive algorithm performance for different average stay durations, internet cafe size = 500 with 95% confidence intervals

Average Stay	Optimal	Competitive		Aggressive Competitive	
2.5 Hour	4895.9	4291.3	87.6% $\pm 0.96\%$	4754.82	97.1% $\pm 0.45\%$
3.5 Hour	6011.6	5083.6	84.6% $\pm 0.84\%$	5696.7	94.8% $\pm 0.66\%$
4.5 Hour	7149.4	5787.6	81.0% $\pm 0.55\%$	6619.7	92.6% $\pm 0.48\%$
6.5 Hour	8696.0	6704.3	77.1% $\pm 0.77\%$	7866.9	90.5% $\pm 0.46\%$
Average			82.6% $\pm 0.78\%$		93.7% $\pm 0.51\%$

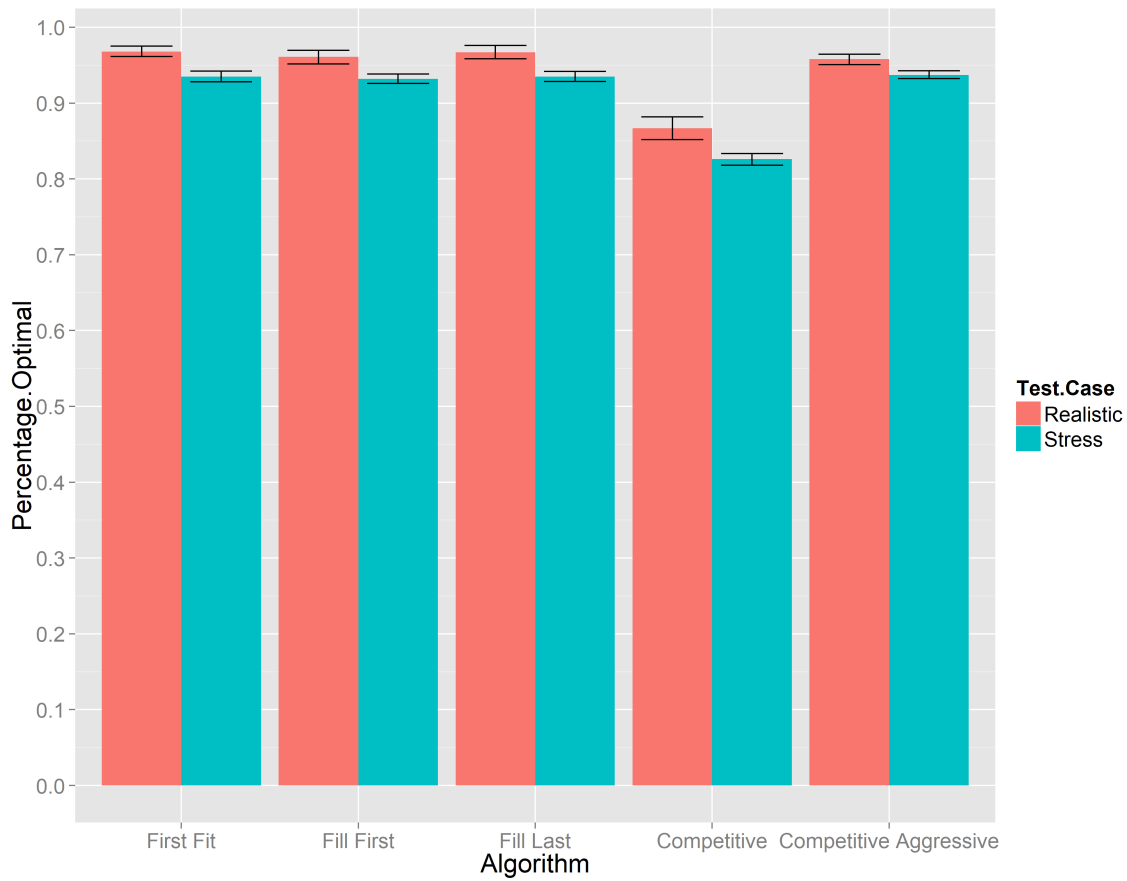


FIGURE 7.1: Percentage From Optimal for Online Algorithms (average for all realistic and stress test cases)

7.3.3 Comparison

Overall both algorithms have very similar performance. The correct choice of algorithm will depend on the user demand for each internet cafe. Figure 7.1 shows percentage difference between the online algorithms averaged over all test cases split by the realistic and stress test sets. It is clear that all algorithms but the standard competitive algorithm can offer profits of 90%+ of the optimal profit with all algorithms performing worse on the stress tests.

In order to illustrate the power of the aggressive competitive algorithm both the greedy and competitive algorithms were used to solve the special test case with a 25 seat internet cafe described in §4.3.6.2. In this internet cafe a set of users arrives who fill the entire internet cafe and stay for 3 hours then while those users would still be present another set of users arrive staying for 5 hours, these more profitable users will have to be rejected if the previous

users were all accepted. All three greedy algorithms have no choice but to accepted all users and fill the internet cafe with the 3 hour users. While the aggressive competitive algorithm strategically rejects users that would fill servers in case of potential future valuable users. Table 7.6 shows the results where the greedy algorithms only obtain 70% of the optimal profit while the aggressive competitive algorithm is able to obtain 89% of the optimal.

TABLE 7.6: Results and Comparison for 20 Seat Internet Cafe

	Optimal	Greedy		Competitive	
25 Seats	116.9	81.9	70.1%	104	89.0%

The results presented indicate that the competitive aggressive algorithm would be a good choice for a cloud-based internet cafe. It offers acceptable profits in the real world test cases while being less prone to error than the greedy algorithms. Results show both the greedy and aggressive competitive algorithms have similar performance in the realistic test cases but the greedy algorithm has poor performance on the special test case. However, the greedy algorithm is the superior choice if the resources are not highly contested or if it is preferable to attempt to place every user who enters the internet cafe. This choice is confirmed by the superior performance of the greedy algorithms in §7.3.1. However, if resources are highly contested then the competitive algorithm will allocate resources to more profitable users as shown in §7.3.2.

Overall both the greedy and competitive algorithms have potential applications depending on the unique usage profile of the internet cafe using the cloud-based system. Next the algorithms are compared with the prebooking system and the offline IP considering the different applications and user demands where each algorithm performs best.

Chapter 8

Discussion and Conclusion

In this chapter the results, strengths, weakness and applications of the algorithms from chapters 5, 6, and 7 are discussed. We also discuss the potential applications for the model presented in chapter 4 for the cloud, internet cafes, educational institutions and businesses.

The offline integer program (Chapter 5), prebooking integer program (Chapter 6), greedy online algorithm (Chapter 7), and competitive online algorithm (Chapter 7) each have their own strengths and weaknesses. All four of these algorithms can be utilised in different ways depending on what an internet cafe is trying to accomplish.

These algorithms are all designed to solve the resource allocation problem in a cloud-based internet cafe. The offline integer program is designed to allocate resources for historical demand. The prebooking integer program is designed for a system where users book in advance for a seat in the internet cafe. The greedy and competitive algorithms are designed to allocate resources to users as they arrive.

These algorithms have applications beyond a cloud-based internet cafe. In particular, they can be applied to graphics-driven cloud applications such as [GaaS](#). Additionally, there is a business application for the cloud-based internet cafe model. A business' workstations, especially high-performance workstations, could be replaced with servers. This architecture

change offers a similar resource efficiency improvement as an internet cafe experiences when switching to a cloud-based internet cafe. Applications that were limited to high-performance workstations could now be accessed via thin-client machines throughout the business and expensive upgrades only affect the (small number of) servers. Universities and schools could similarly benefit in situations where computer labs have been specialised for specific classes. Under a cloud-based model, a lab could supply classes with any computing power required.

8.1 Comparison

The four algorithms each have a different user demand profile and future information availability where they are best utilised. Table 8.1 shows the key differences between the different algorithms concerning solution times, user information needed, and solution quality.

TABLE 8.1: Comparison of algorithm performance

	Solve Time (seconds)	Future User Information	Solution Quality	Solution Period
Offline	600	Perfect	Optimal	All users
Prebooking	120	Perfect up to time horizon	98% Optimal	All users within the time horizon
Greedy	1	None	94.9% Optimal	Current user
Competitive	1	None	94.6% Optimal	Current user

The offline [IP](#) is solved to find the optimal profit for the entire time period in a single 600 second solve. To do this, it requires full information for the entire time period. The offline [IP](#) is useful for finding out what could have been done better using historical data. It can also be used to calculate the number of servers which may be needed to supply a cloud-based internet cafe.

The prebooking [IP](#) is solved to find the optimal profit within a time horizon taking 120 seconds to solve. This solve must occur multiple times to find a solution for the entire time period. Over the entire time period the prebooking [IP](#) on average achieves 98% of the offline optimal for problem instances considered in this thesis.

The greedy and competitive algorithms require no future information and solve every time a new user arrives. Over the entire time period, the algorithms on average earn profits just under 95% of the offline optimal for problem instances considered in this thesis.

The prebooking, greedy, and competitive algorithms are all useful for allocating resources in an online setting within a cloud-based internet cafe. The prebooking algorithm can provision partially into the future and can inform users in advance if they will receive a seat in the internet cafe. In exchange, users must book in advance and are then informed later if they have been accepted.

The greedy algorithm is simple and allocates users as they arrive. This algorithm matches how a traditional internet cafe accepts users. The competitive algorithm offers a way to allocate users as they arrive with the added ability to reject users under certain conditions. The ability to reject is particularly powerful if there are many more users than resources available, making the selection of the most profitable users more desirable.

8.2 Internet Cafe Application

The cloud-based internet cafe requires a hardware, and software setup with an algorithm to allocate the resources.

The process of building a cloud-based internet cafe is described in chapter 3. The results of this chapter indicate that the best hypervisor would be XenServer, with RemoteFX as the remote desktop software. The virtual machines would run Windows 10 with the local thin clients running Ubuntu. A software layer would need to be built to manage the creation of VMs and connect users with their desired service.

The choice of algorithms for allocating server resources depends on the circumstances of the internet cafe. If an internet cafe has a large number of walk-in users which do not often use the full resources available in the internet cafe it is best to use the greedy algorithm. The greedy algorithm will always accept users which, in this situation, is preferable to rejecting

users as all resources are not often utilised. This superior performance is shown in the results presented in §7.3.1, where the greedy algorithm outperforms the competitive algorithm if the internet cafe seldom uses all resources.

Similarly, if an internet cafe has a large number of walk-in users that often exceed the available resources, then it is best to use the competitive algorithm. The competitive algorithm is best because when resources are more scarce the ability to reject users in favour of a more profitable user in the future is beneficial. This superior performance is shown in the results presented in §7.3.2, where the competitive algorithm outperforms the greedy algorithm in situations where the internet cafe has more users than available resources.

Alternatively, if an internet cafe has mostly repeat users with consistent arrival times, then the prebooking system would be the best algorithm as these users would be able to consistently book in advance. The prebooking system obtains the most profit of the three online algorithms but requires the most information. Therefore this algorithm can only be used in a scenario where this extra information can be easily obtained. Given internet cafes are generally more successful when they have repeat customers [23], it is likely many internet cafes have consistent demand from these repeat customers. This consistent demand can be used in a prebooking system allowing the internet cafe to both plan for users and obtain greater profits.

An implemented system for a cloud-based internet cafe is likely to use a prebooking system for repeat customers and allocate walk-in users separately. Placement of walk-in users on the remaining server resources can be managed using either greedy or competitive strategies depending on how busy the internet cafe is on average.

8.3 Graphics Driven Cloud Application

Given the cloud-based nature of the cloud-based internet cafe, a natural application for the algorithms is the allocation of resources in a cloud with vGPUs. These graphics driven clouds run graphics intensive applications including computational GPU services or GaaS.

In a cloud users arrive with “jobs” for the cloud to perform. These jobs consume a number of resources for a period of time similar to users in an internet cafe. For graphics intensive applications the network of the cloud must also be considered as a limited resource. All the algorithms are capable of handling an additional resource in the resource set and, as such, it is feasible that networking could be considering with minimal additional work.

If the cloud has graphics intensive jobs which repeat consistently such as daily solves of problems or updates to calculations, then it would be beneficial to use the offline [IP](#) to allocate these jobs optimally. If demand is uncertain between days but known by users in advance, then the prebooking system could be utilised. On the other hand, uncertain demand such as that experienced in a [GaaS](#) application would be best served by either the greedy or competitive algorithms depending on the resource congestion. The competitive algorithm would be best for highly congested clouds, and the greedy algorithm if the resources are not limiting.

8.4 Business Application

A similar application to the cloud-based internet cafe is the cloud-based workstation. Cloud-based workstations is a system which would replace all desktop workstation computers in a business with [VMs](#) and thin clients.

As in an internet cafe, a business consists of many [PCs](#) which are used by staff on a daily basis. Just like an internet cafe, these are often leased with most staff not utilising the full potential of the [PC](#). Additionally, some businesses may have particular high power workstations used either by a specific staff member or shared by multiple staff. The staff must physically sit at the computer to use this [PC](#). The choice to have a small quantity of high power workstations is often driven by the high cost to license specialist software. The cloud-based approach allows a single instance of the software to run on a [VM](#) which any staff member can access remotely from any desk.

A business is likely to have predictable demand as staff arrive and depart at consistent times and request mostly consistent resources on a day to day basis. If demand is known, then the offline [IP](#) would be able to provide a solution to the resource allocation problem for staff workstations. This consistency means a business could optimally utilise the available resources. In addition, the business would be able to calculate the exact quantity of servers needed for the business by using the offline [IP](#).

The prebooking system would be beneficial for [VMs](#) running limited license software. Staff members could book the [VM](#) in advance and receive confirmation on availability and have resources allocated on a server for the task.

8.5 Education Application

The cloud-based internet cafe model also has applications in educational institutes in the form of cloud-based computer labs. The cloud-based computer lab is a system in which the functionality of all computers in computer labs inside an educational institute would be replaced by functionality provided by [VMs](#) supplied from centralised servers.

In this model computer labs in a university would have no speciality and as such any class could be taught in any computer lab. Typically with non cloud-based computer labs engineering classes need to be taught in engineering computer labs, which have the correct software installed, while science classes require different labs and software. The cloud-based computer labs would have sets of [VMs](#) for engineering classes, science classes, and other classes taken in computer labs. These [VMs](#) could then be accessed from any physical computer lab within the school or university. This flexibility allows the university to teach any class in any computer lab and can allocate labs based on location or size relative to the students and staff rather than by computing power or software installed.

The cloud-based computer lab problem would be solved using the offline [IP](#) because all classes and their details are known in advance, so the [VMs](#) can be allocated using the offline

IP solution. Implementation of the cloud-based computer lab also involves a scheduling optimisation problem of when and where to hold the computer labs. This problem needs to be solved in order to remove any conflicts with other classes, to keep travel times to and from the class low, and ensure the server resources are not over-allocated but also not under-allocated.

The algorithms developed for cloud-based internet cafe resource allocation have shown very similar performance on average in testing but each outperforms the others in specific test sets. These results show specific situations that allow each algorithm to provide the best profits and the consideration of other applications for a cloud-based model where those situations occur.

8.6 Conclusion

This thesis has presented hardware and software setups for a cloud-based internet cafe and four potential algorithms which can be used to allocate resources in different situations. The core application presented in this thesis is the cloud-based internet cafe. However, this research is also applicable to graphics driven clouds, cloud-based business workstations, and cloud-based educational computer labs.

8.6.1 Key Outcomes

The key results of this thesis include the analysis of: 1) server performance for gaming; 2) software options for remotely supplying games; and 3) algorithm development and testing for cloud-based internet cafe resource allocation. Offline and online algorithms have been developed for efficiently allocating users and resources in different internet cafe demand scenarios.

One significant result is the evaluation of the performance of servers and software in a cloud gaming system. In particular the performance of the new vGPU technology and related software.

This thesis tested Nvidia GRID K1 and K2 cards, first generation hardware by Nvidia using vGPU technology. The methodology developed could also be applied to the next generation Nvidia Tesla GPUs. Extensive testing showed that the Nvidia GRID K1 card is only capable of running the lowest requirement games, while the Nvidia GRID K2 card is capable of playing all games as long as the settings are turned down. Testing also showed that adding more CPUs to a VM increases the performance of games although not as significantly as changing a VM from a K1 to K2 GPU.

When starting this research, Citrix XenServer was the only hypervisor to support vGPU technology and was used for all tests in this thesis. The stability of XenServer vGPUs improved significantly over the last few years from version 5.1 to 7. While Citrix XenDesktop was able to manage VMs and supply stable remote desktop connections with vGPUs, it did not support gaming. Instead, it was found that RemoteFX was able to provide a good quality gaming experience, but still requires stable Linux client software.

Algorithms developed for offline resource allocation in a cloud-based internet cafe showed that a cloud-based approach outperforms a traditional zoned internet cafe. Testing with the offline integer program showed an improvement in profits of 4% from having flexible rather than fixed zone sizes. The offline integer program also showed an improvement in resource utilisation from 30% to 70%. The offline IP is the first offline algorithm to solve the VM placement problem with the full complexity of vGPUs.

Algorithms were developed for usage in a cloud-based internet cafe. These online algorithms included a prebooking system, a greedy, and a competitive algorithm.

The prebooking system is useful for allocation of regular users who know when they will arrive in advance. It was shown that with the correct solve time limit, and time horizon the prebooking algorithm can gain over 99% of the offline optimal profit. The number of hours that users must book in advance to achieve the high percentage of offline optimal profit is proportional to the average number of hours users stay.

The greedy algorithm is useful for inconsistent user demand which does not significantly exceed the available server resources. The greedy strategies can achieve on average 95% of the offline optimal profit over the test data sets. All three greedy strategies (first fit, fill first, and fill last) have similar performance on average over all test data sets in this thesis. Greedy algorithms have been used in both cloud scheduling and cloud gaming before [75], but this thesis is the first to schedule VMs utilising vGPUs for cloud gaming.

The competitive algorithm is useful for inconsistent user demand which significantly exceeds the available server resources. The aggressive competitive strategy can achieve 94.5% of the offline optimal on average for all test data sets. The original competitive algorithm only reached 85% of the offline optimal profit. For this reason, an aggressive tilt was added making the competitive algorithm more willing to accept users and allowing it to achieve a much higher percentage of the optimal profit. The improvements of the aggressive competitive algorithm demonstrates a need to tune the competitive algorithms aggressiveness uniquely to each specific problem instance. The competitive algorithm is a significant adaptation of an existing phone routing problem adding a time axis and connected resources within servers. The results give a competitive ratio for the online cloud-based internet cafe problem.

8.6.2 Future Work

Further work needs to be conducted including testing the latest vGPU hardware and testing algorithms with real data from internet cafes. In addition, the entire system needs a wrapper developed which allows resources to be automatically allocated and users to log in and access the VMs easily. Once this is implemented, the system would need to be prototyped in an actual internet cafe.

Newer Nvidia Tesla cards explicitly created for cloud gaming have been released during this thesis. These likely offer significant performance improvements over the Nvidia GRID cards. In particular, the Nvidia Tesla M60 is worth benchmarking as Nvidia claims to use this card for their cloud gaming service.

While this thesis has built realistic data sets driven by surveys and data on games played, it would be useful to build a real-world data set using internet cafe user demands. This dataset would allow a more precise judgement on the applicability of algorithms to real internet cafes especially if coupled with testing on the latest hardware.

In addition, a user interface (UI) and backend need to be developed which can take a users request for a service and run it through one of the algorithms to accept or reject, and automatically create VMs to supply the service. This UI should also act as a wrapper to allow users to log in and connect to their VM within the internet cafe.

The final step in this process would be prototyping of a cloud-based internet cafe in a real internet cafe.

8.6.3 Application

This thesis has presented the blue-print for a cloud-based internet cafe and algorithms for such an internet cafe's operation.

This cloud-based internet cafe replaces the typical computers used in an internet cafe with thin clients which connect remotely to VMs provisioned on servers. These servers make use of vGPU technology to add flexibility when supplying services to internet cafe users. Current internet cafes have improved resource efficiency by splitting the internet cafe into two zones. In a cloud-based internet cafe the services are divided into more (virtual) zones to improve resource utilisation and hence the efficiency of purchased resources. This system has the additional advantage of allowing users to access any service from any seat, allowing the internet cafe to alter the quantity of each service being supplied at any time.

The algorithms presented have shown the ability to efficiently allocate resources for a cloud-based internet cafe with different types of users to maximise the internet cafes profit. The offline IP has provided a benchmark best performance for the cloud-based internet cafe resource allocation problem. The prebooking system is efficient for an internet cafe with repeat

users. The greedy algorithm is efficient for an internet cafe with walk-in users who rarely use all seats in the internet cafe. The competitive algorithm is efficient for an internet cafe with walk-in users who often exceed the number of seats in the internet cafe.

8.6.4 Research Questions

This thesis has answered the fundamental research questions for developing a cloud-based internet cafe:

1. can cloud gaming be combined with internet cafes, to create a new system known as a cloud-based internet cafe, which alleviates the inefficiencies of both internet cafes and cloud gaming?
2. can optimisation algorithms provide methods for the efficient allocation of cloud resources in a cloud-based internet cafe, particularly given the use of GPU resources?

In answer to these research questions:

1. this thesis has shown that cloud gaming can be combined with internet cafes to create a cloud-based internet cafe using vGPU technology. vGPU technology was extensively tested with virtualisation software options which demonstrated the viability of using VMs with vGPUs to provide video games remotely. The combination of an offline integer program developed to maximise the profit of a cloud-based internet cafe and a test data set developed to produce typical internet cafe demand showed that the cloud-based internet cafe model significantly improves resource efficiency and increases profits.
2. algorithms were developed which placed users onto resources in an online manner equivalent to the day to day operation of a cloud-based internet cafe. Three online strategies have been developed and tested each showed the ability to exceed 90% of the profit obtained by the offline optimal (with its proven resource efficiency) and hence, showing

that optimisation algorithms can provide methods for the efficient allocation of cloud resources.

Overall, this thesis has presented a new paradigm for efficiently allocating cloud graphics resources based on user demand. This new paradigm has considered the hardware and software setup, and offline and online resource allocation algorithms. Extensive testing was carried out on both the hardware and resource allocation algorithms using representative internet cafe data sets. Results showed that a cloud-based internet cafe is feasible with the tested hardware and software and that the developed algorithms were capable of efficiently allocating resources including **vGPU**. These algorithms are unique in their ability to handle **vGPU** resources which allows them to be used in a cloud-based internet cafe and the other applications presented.

Appendix A

Configurations

TABLE A.1: K1 server configurations for test set 1

Configuration	vGPU	Number of Low VMs	Number of Medium VMs	Number of High VMs	CPU Usage	RAM Usage	GPU Usage
K1-1	K140	1	0	0	4	6	0.25
K1-2	K140	2	0	0	8	12	0.5
K1-3	K140	3	0	0	12	18	0.75
K1-4	K140	4	0	0	16	24	1
K1-5	K140	1	1	0	10	14	0.75
K1-6	K140	2	1	0	14	20	1
K1-7	K140	0	1	0	6	8	0.5
K1-8	K140	0	2	0	12	16	1

TABLE A.2: K2 server configurations for test set 2

Configuration	vGPU	Number of Low VMs	Number of Medium VMs	Number of High VMs	CPU Usage	RAM Usage	GPU Usage
K2-1	K220	1	0	0	4	6	0.125
K2-2	K220	2	0	0	8	12	0.25
K2-3	K220	3	0	0	12	18	0.375
K2-4	K220	4	0	0	16	24	0.5
K2-5	K220	5	0	0	20	30	0.625
K2-6	K220	6	0	0	24	36	0.75
K2-7	K220	7	0	0	28	42	0.875
K2-8	K220	8	0	0	32	48	1
K2-9	K220	1	1	0	10	14	0.375
K2-10	K220	2	1	0	14	20	0.5
K2-11	K220	3	1	0	18	26	0.625
K2-12	K220	4	1	0	22	32	0.75
K2-13	K220	5	1	0	26	38	0.875
K2-14	K220	6	1	0	30	44	1
K2-15	K220	1	2	0	16	22	0.625
K2-16	K220	2	2	0	20	28	0.75
K2-17	K220	3	2	0	24	34	0.875
K2-18	K220	4	2	0	28	40	1
K2-19	K220	1	3	0	22	30	0.875
K2-20	K220	2	3	0	26	36	1
K2-21	K220	0	1	0	6	8	0.25
K2-22	K220	0	2	0	12	16	0.5
K2-23	K220	0	3	0	18	24	0.75
K2-24	K220	0	4	0	24	32	1
K2-25	K240	1	0	1	12	14	0.625
K2-26	K240	2	0	1	16	20	0.75
K2-27	K240	3	0	1	20	26	0.875
K2-28	K240	4	0	1	24	32	1
K2-29	K240	0	1	1	14	16	0.75
K2-30	K240	0	2	1	20	24	1
K2-31	K240	1	1	1	18	22	0.875
K2-32	K240	2	1	1	22	28	1
K2-33	K240	0	0	1	8	8	0.5
K2-34	K240	0	0	2	16	16	1
K2-35	K240	1	0	0	4	6	0.125
K2-36	K240	2	0	0	8	12	0.25
K2-37	K240	3	0	0	12	18	0.375
K2-38	K240	4	0	0	16	24	0.5
K2-39	K240	1	1	0	10	14	0.375
K2-40	K240	2	1	0	14	20	0.5
K2-41	K240	0	1	0	6	8	0.25
K2-42	K240	0	2	0	12	16	0.5
K2-43	K240	0	3	0	18	24	0.75
K2-44	K240	0	4	0	24	32	1

Bibliography

- [1] Niko Partners. China’s location of gameplay and games hardware report 2013, 2013.
- [2] Anne Sofie Laegran and James Stewart. Nerdy, trendy or healthy? configuring the internet café. *New Media & Society*, 5(3):357–377, 2003.
- [3] Nina Wakeford. The embedding of local culture in global communication: independent internet cafés in london. *New Media & Society*, 5(3):379–399, 2003.
- [4] S. S. Alam, Z. Abdullah, and N. Ahsan. Cyber café usage in Malaysia: An exploratory study. *Journal of Internet Banking and Commerce*, 14(1):1–13, 2009.
- [5] Jennifer Burrell. *Producing the Internet and Development: an ethnography of Internet cafe use in Accra, Ghana*. PhD thesis, The London School of Economics and Political Science (LSE), 2012.
- [6] Craig Glenday. *Guinness World Records*. Jim Pattison Group, 2008.
- [7] Sony. PlayStation Now. <https://www.playstation.com/en-us/explore/psnow/>, 2015.
- [8] NVIDIA. Cloud gaming - gaming as a service (GaaS) — Geforce Now. <https://www.nvidia.com/en-us/geforce/products/geforce-now/>, 2017.
- [9] LiquidSky. LiquidSky. <https://www.liquidsky.com/>, 2015.
- [10] Amazon. Amazon web services. <https://aws.amazon.com/>, 2017.

- [11] Microsoft. Microsoft azure cloud computing platform & services. <https://azure.microsoft.com/en-us/>, 2017.
- [12] Google. Google docs - create and edit documents online, for free. <https://www.google.com/docs/about/>, 2017.
- [13] Netflix. Netflix. <https://www.netflix.com/>, 2017.
- [14] Nelson Ruest and Danielle Ruest. *Virtualization, A Beginner's Guide*. McGraw-Hill, Inc., 2009.
- [15] Charu Chaubal. The architecture of vmware esxi. *VMware White Paper*, 1(7), 2008.
- [16] Citrix. Xenserver - server virtualization and consolidation - citrix - citrix. <https://aws.amazon.com/>, 2017.
- [17] Anthony Velte and Toby Velte. *Microsoft virtualization with Hyper-V*. McGraw-Hill, Inc., 2009.
- [18] Irfan Habib. Virtualization with kvm. *Linux Journal*, 2008(166):8, 2008.
- [19] Minxian Xu, Wenhong Tian, and Rajkumar Buyya. A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience*, 29(12), 2017.
- [20] Nvidia. NVIDIA GRID K1 and K2 graphics-accelerated virtual desktops and applications. <http://www.nvidia.com/content/cloud-computing/pdf/nvidia-grid-datasheet-k1-k2.pdf>, 2013.
- [21] AMD. Professional gpus for servers. <http://www.amd.com/en-us/products/graphics/server>, 2017.
- [22] James Stewart. Cafematics: The cybercafe and the community. *Community Informatics*, pages 320–338, 1999.

- [23] Stein Kristiansen, Bjørn Furuholt, and Fathul Wahid. Internet cafe entrepreneurs: pioneers in information dissemination in indonesia. *The International Journal of Entrepreneurship and Innovation*, 4(4):251–263, 2003.
- [24] Stephen M. Mutula. Cyber caf industry in africa. *Journal of Information Science*, 29(6):489–497, 2003. doi: 10.1177/0165551503296006. URL <https://doi.org/10.1177/0165551503296006>.
- [25] M. Gürol and T. Sevindik. Profile of Internet cafe users in Turkey. *Telematics and Informatics*, 24(1):59–68, 2007.
- [26] Aysegul Yolga Tahiroglu, Gonca G Celik, Mehtap Uzel, Neslihan Ozcan, and Ayse Avci. Internet use among turkish adolescents. *CyberPsychology & Behavior*, 11(5):537–543, 2008.
- [27] Syeda Hina Batool and Khalid Mahmood. Entertainment, communication or academic use? a survey of internet cafe users in lahore, pakistan. *Information Development*, 26(2):141–147, 2010.
- [28] Jenna Burrell. Could connectivity replace mobility? an analysis of internet cafe use patterns in accra, ghana. *Mobile phones: The new talking drums of everyday Africa*, pages 151–169, 2009.
- [29] P Lal, R Malhotra, C Ahuja, and GK Ingle. Internet use among medical students and residents of a medical college of north india. *Indian Journal of Community Medicine*, 31(4):293–294, 2006.
- [30] Peter G Mwesige. Cyber elites: A survey of internet café users in uganda. *Telematics and Informatics*, 21(1):83–101, 2004.
- [31] Hui Wang, Xiaolan Zhou, Ciyong Lu, Jie Wu, Xueqing Deng, and Lingyao Hong. Problematic internet use in high school students in guangdong province, china. *PloS one*, 6(5):e19660, 2011.

- [32] Bin Liang and Hong Lu. Internet development, censorship, and cyber crimes in china. *Journal of Contemporary Criminal Justice*, 26(1):103–120, 2010.
- [33] Esharenana E Adomi, Faith Sarah Omodeko, and Patience Uzezi Otolu. The use of cybercafé at delta state university, abiraka, nigeria. *Library hi tech*, 22(4):383–388, 2004.
- [34] Sarah Lee. Private uses in public spaces: A study of an internet cafe. *New Media & Society*, 1(3):331–350, 1999.
- [35] Nina Wakeford. 11 gender and the landscapes of computing in an internet café. *Virtual geographies: Bodies, space and relations*, page 178, 1999.
- [36] Cormac Herley and Dinei Florencio. How to login from an internet café without worrying about keyloggers. In *Symp. on Usable Privacy and Security*, 2006.
- [37] Dwaine Clarke, Blaise Gassend, Thomas Kotwal, Matt Burnside, Marten Van Dijk, Srinivas Devadas, and Ronald Rivest. The untrusted computer problem and camera-based authentication. In *International Conference on Pervasive Computing*, pages 114–124. Springer, 2002.
- [38] AK Al-Ghamdi, SMA Abdelmalek AM Ashshi, H Faidah, H Shukri, and AA Jiman-Fatani. Bacterial contamination of computer keyboards and mice, elevator buttons and shopping carts. *African Journal of Microbiology Research*, 5(23):3998–4003, 2011.
- [39] Wan-Kuen Jo and Young-Jun Seo. Indoor and outdoor bioaerosol levels at recreation facilities, elementary schools, and homes. *Chemosphere*, 61(11):1570–1579, 2005.
- [40] Yashpalsinh Jadeja and Kirit Modi. Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 877–880. IEEE, 2012.
- [41] David E Williams. *Virtualization with Xen (tm): Including XenEnterprise, XenServer, and XenExpress*. Syngress, 2007.

-
- [42] VMWare. Server virtualization software — vsphere — vmware. <https://www.vmware.com/products/vsphere.html>, 2017.
 - [43] Microsoft. Microsoft management console - overview. <https://www.citrix.com/products/xenserver/>, 2017.
 - [44] Daniel P. Berrang. Virtual machine manager home. <https://virt-manager.org/>, 2017.
 - [45] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), 2012.
 - [46] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open vswitch. In *NSDI*, pages 117–130, 2015.
 - [47] Michael Nelson. Virtual machine migration, January 27 2009. US Patent 7,484,208.
 - [48] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, and Xiaodong Pan. Live virtual machine migration with adaptive, memory compression. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE, 2009.
 - [49] Timothy Wood, Prashant J Shenoy, Arun Venkataramani, Mazin S Yousif, et al. Black-box and gray-box strategies for virtual machine migration. In *NSDI*, volume 7, pages 17–17, 2007.
 - [50] Michael Nelson, Beng-Hong Lim, Greg Hutchins, et al. Fast transparent migration for virtual machines. In *USENIX Annual technical conference, general track*, pages 391–394, 2005.
 - [51] Garth A Gibson and Rodney Van Meter. Network attached storage architecture. *Communications of the ACM*, 43(11):37–45, 2000.

- [52] Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association, 2006.
- [53] Microsoft. Windows storage server overview. [https://technet.microsoft.com/en-us/library/jj643303\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/jj643303(v=ws.11).aspx), 2017.
- [54] Google. Google cloud computing, hosting services & apis — google cloud platform. <https://cloud.google.com/>, 2017.
- [55] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.
- [56] Sipat Triukose, Zhihua Wen, and Michael Rabinovich. Measuring a commercial content delivery network. In *Proceedings of the 20th international conference on World wide web*, pages 467–476. ACM, 2011.
- [57] Daniel J Mendez, Mark D Riggins, Prasad Wagle, and Christine C Ying. System and method for securely synchronizing multiple copies of a workspace element in a network, July 4 2000. US Patent 6,085,192.
- [58] Tom Henderson and Brendan Allen. Vmware view, citrix xendesktop win vdi software shootout. *Network World*, 26:24–30, 2009.
- [59] Citrix. Xenapp and xendesktop - virtual apps and desktops. <https://www.citrix.com/products/xenapp-xendesktop/>, 2017.
- [60] VMWare. Horizon 7 — virtual desktop infrastructure — vdi — vmware. <https://www.vmware.com/products/horizon.html>, 2017.
- [61] Microsoft. Welcome to remote desktop services. <https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/welcome-to-rds>, 2017.

-
- [62] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.
- [63] Kris Gray. *Microsoft DirectX 9 programmable graphics pipeline*. Microsoft Press, 2003.
- [64] Microsoft. Microsoft remotefx. [https://technet.microsoft.com/en-us/library/ff817578\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/ff817578(v=ws.10).aspx), 2017.
- [65] Wei Cai, Min Chen, and Victor Leung. Toward gaming as a service. *IEEE Internet Computing*, 18(3):12–18, 2014.
- [66] Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 1269–1272, 2011.
- [67] Chun-Ying Huang, Kuan-Ta Chen, De-Yu Chen, Hwai-Jung Hsu, and Cheng-Hsin Hsu. GamingAnywhere: the first open source cloud gaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(1s):10:1–10:25, 2014.
- [68] NVIDIA. Cloud gaming - gaming as a service (GaaS) — NVIDIA GRID. <http://www.nvidia.com/object/cloud-gaming.html>, 2014.
- [69] Dapeng Wu, Yiwei Thomas Hou, and Ya-Qin Zhang. Transporting real-time video over the Internet: Challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1877, 2000.
- [70] Sari Järvinen, Jukka-Pekka Laulajainen, Tiia Sutinen, and Sami Sallinen. QoS-Aware real-time video encoding - how to improve the user experience of a gaming-on-demand service. In *Proceedings of the 3rd Annual IEEE Consumer Communications and Networking Conference (CCNC)*, volume 2, pages 994–997, 2006.

- [71] Pedro Casas, Michael Seufert, Sebastian Egger, and Raimund Schatz. Quality of experience in remote virtual desktop services. In *Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management*, pages 1352–1357, 2013.
- [72] Chun-Ying Huang, De-Yu Chen, Cheng-Hsin Hsu, and Kuan-Ta Chen. GamingAnywhere: an open-source cloud gaming testbed. In *Proceedings of the 21st ACM International Conference on Multimedia*, pages 827–830, 2013.
- [73] Zhengwei Qi, Jianguo Yao, Chao Zhang, Miao Yu, Zhizhou Yang, and Haibing Guan. VGRIS: Virtualized GPU resource isolation and scheduling in cloud gaming. *ACM Transactions on Architecture and Code Optimization (TACO)*, 11(2):17:1–17:25, 2014.
- [74] Chao Zhang, Jianguo Yao, Zhengwei Qi, Miao Yu, and Haibing Guan. vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):3036–3045, 2014.
- [75] Yusen Li, Xueyan Tang, and Wentong Cai. On dynamic bin packing for resource allocation in the cloud. In *Proceedings of the 26th ACM symposium on Parallelism in algorithms and architectures*, pages 2–11. ACM, 2014.
- [76] Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, New York, NY, 2013. ISBN 978-1-4419-7997-1. doi: 10.1007/978-1-4419-7997-1_35. URL https://doi.org/10.1007/978-1-4419-7997-1_35.
- [77] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Introduction to np-completeness of knapsack problems. In *Knapsack problems*, pages 483–493. Springer, 2004.
- [78] Edward G Coffman, Jr, Michael R Garey, and David S Johnson. Dynamic bin packing. *SIAM Journal on Computing*, 12(2):227–258, 1983.
- [79] William Leinberger, George Karypis, and Vipin Kumar. Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In *Parallel*

- Processing, 1999. Proceedings. 1999 International Conference on*, pages 404–412. IEEE, 1999.
- [80] Joseph Y Hui. Resource allocation for broadband networks. *IEEE Journal on selected areas in communications*, 6(9):1598–1608, 1988.
- [81] Peter Mell, Tim Grance, et al. *The NIST definition of cloud computing*. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.
- [82] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, 1988.
- [83] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming*. Princeton university press, 2015.
- [84] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.
- [85] Ragunathan Rajkumar, Chen Lee, John Lehoczky, and Dan Siewiorek. A resource allocation model for qos management. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 298–307. IEEE, 1997.
- [86] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.
- [87] Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 374–382. IEEE, 1995.
- [88] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, and Marco Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

-
- [89] XiaoShan He, XianHe Sun, and Gregor Von Laszewski. Qos guided min-min heuristic for grid task scheduling. *Journal of Computer Science and Technology*, 18(4):442–451, 2003.
- [90] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.
- [91] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.
- [92] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1998.
- [93] Stuart Mitchell, Michael OSullivan, and Iain Dunning. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf, 2011.
- [94] Gurobi Optimization. Gurobi Optimizer. <http://www.gurobi.com>, 2015.
- [95] Robin Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.
- [96] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.
- [97] IBM ILOG CPLEX. V12. 1: Users manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.
- [98] John Forrest and Robin Lougee-Heimer. Cbc user guide. In *Emerging Theory, Methods, and Applications*, pages 257–277. INFORMS, 2005.
- [99] Bernhard Meindl and Matthias Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 2012.

- [100] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimal virtual machine placement across multiple cloud providers. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 103–110. IEEE, 2009.
- [101] Jing Xu and Jose AB Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 179–188. IEEE Computer Society, 2010.
- [102] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, and Naixue Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196, 2013.
- [103] Fangzhe Chang, Jennifer Ren, and Ramesh Viswanathan. Optimal resource allocation in clouds. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 418–425. IEEE, 2010.
- [104] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.
- [105] M. Y. Kovalyov, C.T. Ng, and T. C. E. Cheng. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research (EJOR)*, 178(2):331–342, 2007.
- [106] A. W. J. Kolen, J. K. Lenstra, C. Papadimitriou, and F. C. R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54:530–543, 2007.
- [107] G. J. Woeginger. Online scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.
- [108] R. Canetti and S. Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.
- [109] L. Epstein and A. Levin. Improved randomized results for the interval selection problem. *Theoretical Computer Science*, 411(34–36):3129–3135, 2010.

-
- [110] S. P. Y. Fung, C. K. Poon, and F. Zheng. Online interval scheduling: Randomized and multiprocessor cases. *Journal of Combinatorial Optimization*, 16(3):248–262, 2008.
- [111] H. Miyazawa and T. Erlebach. An improved randomized online algorithm for a weighted interval selection problem. *Journal of Scheduling*, 7(4):293–311, 2004.
- [112] S. S. Seiden. Randomized online interval scheduling. *Operations Research Letters*, 22(4-5):171–177, 1998.
- [113] R. J. Lipton and A. Tomkins. Online interval scheduling. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 302–311, 1994.
- [114] M. Bender, C. Thielen, and S. Westphal. Online interval scheduling with bounded number of failures. *Journal of Scheduling (online first)*, :1–15, 2017. doi: 10.1007/s10951-016-0506-9.
- [115] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proceedings of the 34th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 32–40, 1993.
- [116] J. Aspnes, Y. Azar, , A. Fiat, S. A. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.
- [117] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. Online load balancing of temporary tasks. In *Proceedings of the 3rd Workshop on Algorithms and Data Structures (WADS)*, volume 709 of *LNCS*, pages 119–130, 1993.
- [118] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the 23rd ACM Symposium on the Theory of Computing (STOC)*, pages 101–111, 1991.
- [119] S. Plotkin and D. Shmoys É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 495–504, 1991.

-
- [120] F. Shahrokhi and D. Matula. The maximum concurrent flow problem. *Journal of the ACM*, 37(2):318–334, 1990.
- [121] Thomas L. Casavant and Jon G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, 1988.
- [122] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: Fair scheduling for distributed computing clusters. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP)*, pages 261–276, 2009.
- [123] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking*, 21(6):1785–1798, 2013.
- [124] Vijay Subramani, Rajkumar Kettimuthu, Srividya Srinivasan, and Ponnuswamy Sadayappan. Distributed job scheduling on computational grids using multiple simultaneous requests. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 359–366, 2002.
- [125] Zoltán Ádám Mann. Allocation of virtual machines in cloud data centers a survey of problem models and optimization algorithms. *ACM Computing Surveys (CSUR)*, 48(1):11, 2015.
- [126] Alireza Sadeghi Milani and Nima Jafari Navimipour. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications*, 71:86–98, 2016.
- [127] Kevin Mills, James Filliben, and Christopher Dabrowski. Comparing vm-placement algorithms for on-demand clouds. In *Cloud Computing Technology and Science (Cloud-Com), 2011 IEEE Third International Conference on*, pages 91–98. IEEE, 2011.

- [128] Abhishek Gupta, Laxmikant V Kale, Dejan Milojicic, Paolo Faraboschi, and Susanne M Balle. Hpc-aware vm placement in infrastructure clouds. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 11–20. IEEE, 2013.
- [129] Nicolo Maria Calcavecchia, Ofer Biran, Erez Hadad, and Yosef Moatti. Vm placement strategies for cloud scenarios. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 852–859. IEEE, 2012.
- [130] Chris Hyser, Bret McKee, Rob Gardner, and Brian J Watson. Autonomic virtual machine placement in the data center. *Hewlett Packard Laboratories, Tech. Rep. HPL-2007-189*, 189, 2007.
- [131] William Leinberger, George Karypis, and Vipin Kumar. Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In *Parallel Processing, 1999. Proceedings. 1999 International Conference on*, pages 404–412. IEEE, 1999.
- [132] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor CM Leung, and Cheng-Hsin Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:7605–7620, 2016.
- [133] Yusen Li, Xueyan Tang, and Wentong Cai. Play request dispatching for efficient virtual machine usage in cloud gaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2052–2063, 2015.
- [134] Hua-Jun Hong, De-Yu Chen, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. Placing virtual machines to optimize cloud gaming experience. *IEEE Transactions on Cloud Computing*, 3(1):42–53, 2015.
- [135] David Finkel, Mark Claypool, Sam Jaffe, Thinh Nguyen, and Brendan Stephen. Assignment of games to servers in the onlive cloud game system. In *Network and Systems Support for Games (NetGames), 2014 13th Annual Workshop on*, pages 1–3. IEEE, 2014.

-
- [136] Rubicon Communications. pfsense - world's most trusted open source firewall. <https://www.pfsense.org/>, 2017.
- [137] Klappenbach Michael. Understanding and optimizing video game frame rates. <https://www.lifewire.com/optimizing-video-game-frame-rates-811784>, 2018.
- [138] Valve. Steam in-home streaming. <http://store.steampowered.com/streaming/>, 2017.
- [139] Futuremark. 3DMark. <http://www.3dmark.com/>, 2015.
- [140] AMD Raptr. Most played PC games: January 2015. <http://caas.raptr.com/most-played-pc-games-january-2015-new-year-same-games/>, 2015.
- [141] AMD Raptr. Most played PC games: November 2015. <http://caas.raptr.com/most-played-games-november-2015/>, 2015.
- [142] I. Hamling, M. O'Sullivan, C. Walker, and C. Thielen. Improving resource efficiency in internet cafes by virtualization and optimal user allocation. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 26–34, Dec 2015. doi: 10.1109/UCC.2015.17.
- [143] George B Dantzig and Philip Wolfe. The decomposition algorithm for linear programs. *Econometrica: Journal of the Econometric Society*, pages 767–778, 1961.
- [144] Michael O'Sullivan, Qi-Shan Lim, Cameron Walker, Iain Dunning, and Stuart Mitchell. Dippy: a simplified interface for advanced mixed-integer programming. 2012.