# Predictive Dynamic Resource Allocation for Web Hosting Environments

by

**Mohammed A. AL Ghamdi**

A thesis submitted to The University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

**Doctor of Philosophy**

**Department of Computer Science**

The University of Warwick

February 2012

THE LIBRARY
Tel: +44 24 76523523
Fax: +44 24 76524211

AUTHOR: **Mohammed A. Al Ghamdi** DEGREE: **Ph.D.**

TITLE: **Predictive Dynamic Resource Allocation for Web Hosting Environments**

DATE OF DEPOSIT: .....

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I agree that this thesis may be photocopied (single copies for study purposes only).

Theses with no restriction on photocopying will also be made available to the British Library for microfilming. The British Library may supply copies to individuals or libraries. subject to a statement from them that the copy is supplied for non-publishing purposes. All copies supplied by the British Library will carry the following statement:

> "Attention is drawn to the fact that the copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's written consent."

AUTHOR'S SIGNATURE: .....

---

USER'S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE SIGNATURE ADDRESS

.....

.....

.....

.....

.....

THE UNIVERSITY OF WARWICK, COVENTRY CV4 7AL

# Abstract

E-Business applications are subject to significant variations in workload and this can cause exceptionally long response times for users, the timing out of client requests and/or the dropping of connections. One solution is to host these applications in virtualised server pools, and to dynamically reassign compute servers between pools to meet the demands on the hosted applications. Switching servers between pools is not without cost, and this must therefore be weighed against possible system gain.

This work is concerned with dynamic resource allocation for multi-tiered, cluster-based web hosting environments. Dynamic resource allocation is reactive, that is, when overloading occurs in one resource pool, servers are moved from another (quieter) pool to meet this demand. Switching servers comes with some overhead, so it is important to weigh up the costs of the switch against possible system gains. In this thesis we combine the *reactive* behaviour of two server switching policies – the Proportional Switching Policy (PSP) and the Bottleneck Aware Switching Policy (BSP) – with the *proactive* properties of several workload forecasting models.

We evaluate the behaviour of the two switching policies and compare them against static resource allocation under a range of reallocation intervals (the time it takes to switch a server from one resource pool to another) and observe that larger reallocation intervals have a negative impact on revenue. We also construct model- and simulation-

based environments in which the combination of workload prediction and dynamic server switching can be explored. Several different (but common) predictors – Last Observation (LO), Simple Average (SA), Sample Moving Average (SMA) and Exponential Moving Average (EMA), Low Pass Filter (LPF), and an AutoRegressive Integrated Moving Average (ARIMA) – have been applied alongside the switching policies. As each of the forecasting schemes has its own bias, we also develop a number of meta-forecasting algorithms – the Active Window Model (AWM), the Voting Model (VM), the Selective Model (SM), the Dynamic Active Window Model (DAWM), and a method based on Workload Pattern Analysis (WPA). The schemes are tested with real-world workload traces from several sources to ensure consistent and improved results. We also investigate the effectiveness of these schemes on workloads containing extreme events (e.g. flash crowds). The results show that workload forecasting can be very effective when applied alongside dynamic resource allocation strategies.

to my family

with love and gratitude

# Acknowledgements

I am grateful to many people for their assistance, advice, guidance and friendship during the course of this work. To my supervisor, Prof. Stephen Jarvis, who first guided me to this research area, and giving me the opportunity to work in the Performance Computing and Visualisation Group at Warwick and providing a never-ending source of optimism, good-will and guidance. I am truly grateful for his support, advice and encouragement.

I am grateful to Dr Ligang He for acting as my second supervisor, particularly for his advice during the years of my degree. A special vote of thanks should go to Dr Adam Chester for his relentless hard work, support and the many hours of discussion during our research collaboration. It is a pleasure to acknowledge the many members and colleagues of the Performance and Visualisation Group both past and present including, Dr Simon Hammond (Sandia National Laboratories, USA), Dr Gihan Mudalige (Oxford University, UK), Dr Matthew Leeke, Dr Nathan Griffiths, Dr Arshad Jhumka, Malik Awan, James Davis, Henry Franks, John Pennycook, Oliver Perks, Jessica Smith, Steven Wright (University of Warwick, UK) and last but by no means least, Dr James Xue (Northampton University, UK), who have helped in countless ways over the course, particularly for his advice during the early years of my degree.

Finally, I would like to express my heart-felt gratitude to my family for a constant

source of love, concern, support and strength all these years.

# Declarations

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work described in this thesis has been undertaken by myself except where otherwise stated. Portions of this work have been published in the following publications:

- **M. Al-Ghamdi**, A.P. Chester, J.W.J. Xue, S.A. Jarvis, The Effect of Server Reallocation Time in Dynamic Resource Allocation, UK Performance Engineering Workshop 2009, 6-7th July, 2009, Leeds, UK.

- **M. Al-Ghamdi**, A.P. Chester, S.A. Jarvis, Predictive and Dynamic Resource Allocation for Enterprise Applications, IEEE International Conference on Scalable Computing and Communications, 29 June - 01 July 2010, Bradford, UK.

- **M. Al-Ghamdi**, A.P. Chester, L. He, S.A. Jarvis, Dynamic Resource Allocation and Active Predictive Models for Enterprise Applications. In Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011), 7-9th May, 2011, Noordwijkerhout, The Netherlands.

- A.P. Chester, M. Leeke, **M. Al-Ghamdi**, A. Jumka, S.A. Jarvis, A Modular Failure-Aware Resource Allocation Architecture for Cloud Computing. In: UK

Performance Engineering Workshop 2011, 7-8 July, 2011, Bradford, United Kingdom.

- A.P. Chester, M. Leeke, **M. Al-Ghamdi**, A. Jumka, S.A. Jarvis, A Framework for Data Center Scale Dynamic Resource Allocation Algorithms. 11th IEEE International Conference on Scalable Computing and Communications, 30 August - 2 September 2011, Pafos (Paphos), Cyprus.

- **M. Al-Ghamdi**, A.P. Chester, L. He, S.A. Jarvis, Dynamic Active Window Management: A method for improving revenue generation in Dynamic Enterprise Systems. 11th IEEE International Conference on Scalable Computing and Communications, 30 August - 2 September 2011, Pafos (Paphos), Cyprus.

- **M. Al-Ghamdi**, A.P. Chester, L. He, S.A. Jarvis, Dynamic Resource Allocation for Multi-Tiered, Cluster-Based Web Hosting Environments. Lecture Notes in Business Information Processing (LNBIP) published by Springer-Verlag, 2012.

# Sponsorship and Grants

This work is supported in part by the UK Engineering and Physical Science Research Council (EPSRC) contract number EP/C538277/1, “Dynamic Operating Policies for Commercial Hosting Environments”. This project involved collaboration with IBM, HP Labs, the University of Newcastle and the National Business to Business Centre.

There has also been additional collaboration with Deutsche Bank in the form of capacity planning for a large high-volume application.

# Abbreviations

**PDRAS** - Pre-defined resource allocation system

**DRAS** - Dynamic resource allocation system

**QNM** - Queuing network model

**QoS** - Quality of service

**SLA** - Service level agreement

**QNM** - Queuing network model

**CA** - Convolution algorithms

**MVA** - Mean value analysis

**AMVA** - Approximate mean value analysis

**PSP** - Proportional switching policy

**BSP** - Bottleneck-aware switching policy

**AC** - Admission control

**LO** - Last Observation

**SA** - Simple Algorithm

**SMA** - Sample Moving Average

**EMA** - Exponential Moving Average

**LPF** - Low Pass Filter

**ARIMA** - Autoregressive Integrated Moving Average

**MSE** - Mean Square Error

**MAPE** - Mean Average Percentage Error

**MAD** - Mean Absolute Deviation

**CFE** - Cumulative sum of Forecast Error

**AWM** - Active Window Model

**VM** - Voting Model

**SM** - Selective Model

**DAWM** - Dynamic Active Window Model

**WPA** - Workload Pattern Analysis

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The rapid expansion of the Internet and its applications has forced both industrial and academic researchers to focus their attention on ensuring that these applications deliver appropriate levels of service to their customers without delay or errors. E-Business applications for on-line banking or on-line retail are examples of such Internet applications which are attracting people in their millions to use on-line services of this kind.

Internet workload has been shown to increase significantly with this huge expansion in Internet applications and their uses. Dynamic resource allocation systems (DRAS) play a crucial role in such environments.

Hosting these application is a difficult task. Internet hosting organisations must balance the amount of hardware that they purchase: too few servers may mean that customers requests are dropped or responded to with significant delay; too many servers may mean that during quieter periods, many of the servers go unused.

Pre-defined (or static) resource allocation systems (PRAS), where the available resources are allocated between the different applications at the design stage, are clearly not flexible enough to serve highly variable user demand. DRAS systems are not however cost free and require time to detect and reconfigure to more suitable system arrangement. Added to this, the application (e.g. web service) will not be available

during the reconfiguration process until the new configuration is again successfully deployed. The up-side is that dynamic resource allocation systems can deal with huge variations in application demands, as the available resources are always re-allocated between different applications depending on the demand on these resources from the different applications.

The online applications of interest in this thesis usually employ multi-tiered architectures (*web tier*, *application tier*, and *data-persistence tier*) and are typically hosted on Internet hosting platforms. Each of these online applications usually have separate service level agreements (SLA). These agreements specify the level of service that will be delivered to the customer by the application, based on a specific target for performance (e.g. response time for a given application) and availability (e.g. whether a request is serviced). SLAs also define the penalties that should be paid if such targets are not met or if the level of the delivered service is not that agreed between the application provider and the customer.

## 1.1 Motivation and Problem Statement

Internet hosting centres host multiple Internet applications and require multiple resources to be shared between these different applications. It has been shown that pre-allocating resources at the design stage may affect the system performance, as resources will be wasted in the situation where the demand is low and may the resources requirements exceed the capacity of the system when the demand is high. As a result, dynamic resource allocation systems may be applied in order to improve the total system revenue (for the provider) and to enhance the performance of the whole system (for the customer).

Internet applications should deliver their services within a suitable time without

unnecessary delay which may occur as a result of increasing the number of accesses to that application. In other words, expanding the popularity of an Internet application is not cost free, as more expense is necessary for upgrading and developing the application to maintain the continuing requirements. Improving the application in order to meet the customers' requirements needs to be considered.

Improving service delivery can be done either by studying similar applications that offer similar services or by monitoring the application for a specific period in order to observe the application's behaviour, which usually gives a clear picture as to how the application is used and how it can be developed in the future. Thus, capacity planning and workload forecasting play a crucial role in mitigating such issues [53].

## 1.2 Thesis Contributions

The principal contributions of this thesis are as follows:

- We study the impact of *server switching time* in distributed and dynamic enterprise systems. The switching time is defined as the time taken to reallocate servers between applications. Our aim is to investigate the link between switching time and total system performance, as well as how the switching policies themselves behave with changeable switching times. For this purpose, we integrate two well known switching policies – the Proportional Switching Policy (PSP) and the Bottleneck-aware Switching Policy (BSP) – with variable switching times in a test system. Experiments are conducted on synthetic workloads and we draw conclusions as to the suitability of the switching policies in different practical settings (see Chapter 3).

- We construct a model-based environment in which the combination of work-

load prediction and dynamic server switching can be explored. A multi-tiered, cluster-based, multi-server solution is modelled, which provides bottleneck identification through the use of convex polytopes and also employes admission control. A workload model is also constructed from the characterisation of real data from several different sources. Several schemes for workload prediction have been introduced including: Last Observation (LO), Simple Algorithm (SA), Sample Moving Average (SMA), Exponential Moving Average (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average (ARIMA). A comparison between the forecast accuracy of these schemes *in combination with dynamic server switching* is conducted using several metrics – Mean Square Error (MSE), Mean Average Percentage Error (MAPE), Mean Absolute Deviation (MAD) and the Cumulative sum of Forecast Error (CFE) (see Chapter 4).

- We extend our models in order to address issues that arise when a single forecasting model is used, as each of the forecasting schemes is shown to have its own bias. As a result, three different meta-forecasting algorithms are developed – Active Window Model (AWM), Voting Model (VM), and Selective Model (SM). In the first model (AWM), data points are collected during an active window, and a predictor which deliveries the best revenue for the last active window, is employed. All predictors are used in the voting model (VM) and a server switch is enacted if the majority vote requires. The selective model (SM) applies the best predictor from the last time period to the next time period (see Chapter 5).

- We further extend the Active Window Model (AWM) to a Dynamic Active Window Model (DAWM), where the size of the active window for collecting data points for all predictors is not constant. The window size is calculated based on

either a burstiness factor – the window size decreases when the workload becomes more bursty or based on the correlation between the requests arrival rate and the mean arrival rate for incoming requests to the system. In addition to this, we introduce a historical prediction model – the Workload Pattern Analysis (WPA) model which exploits the periodicity of web traffic to predict workload where the predicted number of requests in this model is related to the previous number of requests found at the same time of day (e.g. the number of requests at mid-night for a specific day is related to the number of requests at midnight recorded during the previous week). All schemes are tested on real-world workloads and also workloads containing extreme events (see Chapter 6).

## 1.3 Thesis Limitations

There are number of important factors – system performance, resource reallocation mechanisms, fault tolerance, and quality of service – that play a crucial role in ensuring that the highest level of success in dynamic resource allocation systems are met. Performance of the system is a significant issue that should be considered in such environments as the ideal target for providing such model is to improve the overall system performance.

How and when to reallocate resources between different applications hosted in an Internet hosting centre is also important and should be studied in such environments. This thesis focus on these two factors through the construction of theoretical models and simulations, where the resources are reassigned between the hosted applications in order to improve the total performance of the system.

There are several other areas that this thesis does not cover. Fault tolerance is one of these many angles that need to be considered in such an environment. The failure

rate of resources could also be considering when developing new dynamic resource allocation policies.

In this thesis, switching servers are considered only between the same tier (e.g. servers from the application tier can be only moved to the application tier of another resource pool). Therefore switching servers between different tiers (e.g. servers from a web tier on a quieter pool are moved to the application tier of another pool) is work to be considered in further research.

## 1.4 Thesis Overview

This chapter detailed the underlying goals, open questions and motivations for the research presented in this dissertation. The remainder of this thesis is organised as follows:

In Chapter 2, background research is presented. This includes different techniques that are applied to improve system performance such as bottleneck and admission control. Dynamic server switching polices are also introduced in order to frame the subsequent research.

Chapter 3 presents the system model that is used in this thesis, where a multi-tiered, cluster-based, multi-server solution is modelled, which contains bottleneck identification through the use of convex polytopes and admission control. The impact of the switching time on such an environment is detailed in this chapter. Experiments are conducted using two inversely proportional workloads.

In Chapter 4, the forecasting of real-world Internet traces is combined together with

dynamic resources allocation. The predictors that are applied here include: Last Observation (LO), Simple Algorithm (SA), Sample Moving Average (SMA), Exponential Moving Average (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average (ARIMA). As each of these forecasting schemes has its own bias, three different meta-forecasting models – Active Window Model (AWM), Voting Model (VM), and Selective Model (SM) – are developed. This is documented in Chapter 5.

A new adaptive predictive model and a historical prediction model are developed in Chapter 6. Experiments are conducted based on both real-world, highly-variable workload traces, and also synthetic workloads with extreme events.

Finally, Chapter 7 provides a summary of the research presented in the thesis and presents future directions for this research.

# Chapter 2

# Background Research

An Internet application is defined as an application that is delivered to an Internet-based user from a server over the Internet. An Internet application is often named based on the services that are provided e.g. when the provided services to the user are related to education or learning purposes, then the Internet application is commonly defined as an e-learning application. When the service is related to financial services e.g. bank transactions, then the system is called e-Banking or e-Business application. Internet hosting centres are often used for the cost-effective hosting of such applications. In this thesis we focus on e-Business applications and how to maximise their performance (specifically revenue generation).

e-Business applications for on-line banking or on-line retail are typically hosted on Internet hosting platforms. These platforms usually employ multi-tiered architectures that are the norm in today's enterprise web applications [36]. For multi-tiered web applications, it is common to have three tiers including a client-facing web tier, an application tier, and a back-end data-persistence tier. The first tier, known as a client-facing web tier, is used for processing the HTTP requests coming from the client/user. It can be also used for security purposes e.g. verifying a user name and password when it is possible. This tier is also used for sending back the response to the client/user. The

application tier is responsible for receiving the request from the first tier and processing it and then sending it either back to the first tier or to the third tier which is known as a back-end data-persistence tier. This third tier is usually comprised of a relational database management system (RDBMS).

This three-tier architecture (see Figure 2.1 [10]) improves the performance of web applications by separating the roles between the different tiers. This means that modification of each tier can be conducted independently and they can be hosted independently on different server architectures. Scalability is also enhanced using such architectures as more application servers can be added without affecting the performance of the system. Further, maintaing the components of the systems can be performed separately without adversely effecting the systems performance.

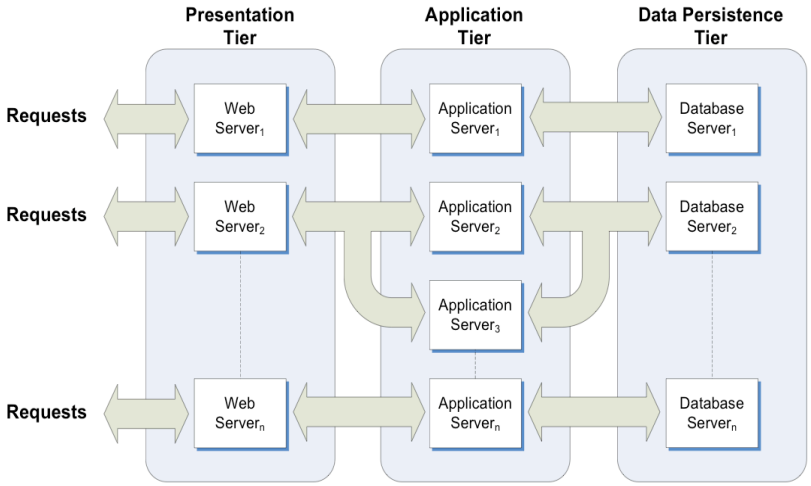


Figure 2.1: Multiple (3-tier) application architecture [10]

## 2.1 Service Level Agreements (SLAs)

IT companies usually use a third party service provider in order to manage and control the Internet resources that are provided to their customers. The relationship between the customers and the service providers is based on a contract that explains the costs of using the services and the penalties that should be paid when the service provider fails to provide sufficient service to their customer. This contract is known as the Service Level Agreement (SLA).

In other words, a SLA is defined as the level of service agreed between the client and the hosting centre and may include performance and availability targets, with penalties to be paid if such targets are not met. It is in the interests of the service hosting centre to ensure that its SLAs are met so as to maximise its revenue, whilst at the same time ensuring that its resources are well utilised and that extra capacity is not provided without need.

## 2.2 Resource Management

e-Businesses are usually hosted through multiple applications and systems, each of which is allocated with a sufficient amount of resources which are usually prescribed during the design phase.

With the significant expansion of the Internet and its applications, organisations are all too aware of increasing server's capacity frequently [56]. Constantly upgrading resources in this way is a naive solution, as increasing the server's capacity requires time and is costly, it is also the case that the server itself will be unavailable until the upgrading process is done.

Clusters of servers may be used to provide high availability and reliability, as a

collection of servers may be connected via a LAN to serve such an environment [55]. The enterprise applications described in this thesis are distributed on high-availability, high-performance clusters of servers.

In such environment when a single server is down the whole application will continue providing its services (since it is hosted on more than one server); maintenance can be conducted on the faulty server in order to resolve this issue, while the remaining servers continue to host application requests. Therefore, the operation of the application will not be affected by the dropping of a single server; the performance of the application may be impacted however.

### 2.2.1 Pre-defined resource allocation system (PRAS)

As stated above, the resources for different applications are usually allocated during the design phase through the process of system capacity planning. Because of the volatile nature of the Internet demand e.g. the visits to Internet applications may vary dramatically over a specific period. In such an environment it is difficult to meet the huge and sudden variation in Internet demand as the configuration (resources) is constant and will never change during the deployment phase.

It has been shown that assigning a fixed number of servers to a resource pool is sub-optimal: in many cases resources lay unused and during peak demand there are insufficient resources to service all requests [10].

### 2.2.2 Dynamic resource allocation system (DRAS)

Allocating resources dynamically during the implementation phase of the system into a more beneficial configuration, based on the variation of demand, is an attractive approach that can be used in order to maximise system performance. That is, when

overloading occurs in one resource pool, servers are moved from another (quieter) pool to meet the increased demand (see Figure 2.2).

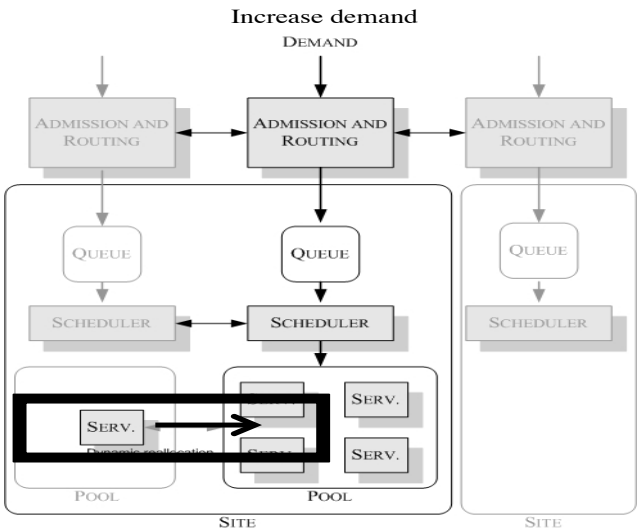


Figure 2.2: Dynamic Resource Allocation over Increased Demand

The switching process between the resource pools can be conducted using different policies to ensure that the resources are reallocated into the most favourable configuration. The most beneficial policy can be determined by comparing the total revenue achieved from reassigning the new resources. In this thesis two main switching policies – the *Proportional Switching Policy* (PSP) and the *Bottleneck-aware Switching Policy* (BSP) – are used, which were developed in [54].

These two policies (PSP and BSP) are reused again in this thesis, as the main aim of thesis is to examine the effectiveness of the switching policies alongside the prediction algorithms, as opposed to developing new switching policies. Added to this, these two policies have been developed as a result of previously funded EPSRC e-Science research (see [54] for more details), and it is our aim to evaluate these in a broader context. The input parameters to both policies (PSP and BSP) are summarised in table

Table 2.1: Notation used in the system model

| Symbol | Description |
|---|---|
| $N$ | Number of service stations in QN |
| $m_i$ | Number of servers at station $i$ |
| $R$ | Number of job classes in QN |
| $K_{ir}$ | Number of class-$r$ job at station $i$ |
| $S_{ir}$ | Service time of job class-$r$ at station $i$ |
| $v_{ir}$ | Visiting ratio of job class-$r$ at station $i$ |
| $\phi_r$ | Revenue of each class-$r$ job |
| $t_s$ | Server switching time |
| $t_d$ | Switching decision interval time |

2.1.

### 2.2.3 Proportional Switching Policy

The proportional switching policy used here is shown in Algorithm 1 and was first presented in [54]. This policy works by allocating servers at each tier proportionally according to workload, subject to an improvement in revenue. The policy is based on the workload as serves are reallocated between the two applications according to the workload. After that, the obtained and lost revenue regarding to the new configurations are computed for both applications; the servers are switched between the applications if the obtained revenue is greater than the lost revenue, otherwise the configuration remains the same.

### 2.2.4 Bottleneck-aware Switching Policy

There are some factors that may affect the system performance e.g. workload mix and revenue contribution from individual classes of job in different pools. The second algorithm which is used in this work is the bottleneck-aware switching policy (BSP), which overcomes these factors in order to obtain improved performance results (see

**Algorithm 1** Proportional switching policy

**Input:** $N$, $m_i$, $R$, $K_{ir}$, $S_{ir}$, $v_{ir}$, $\phi_r$ , $t_s$, $t_d$
**Output:** Server configuration
**for** each $i$ **in** $N$ **do**
  $m_i^1/m_i^2 = K^1/K^2$
**end for**
calculate $V_{loss}$ and $V_{gain}$ using eq. 3.9 and eq. 3.10 (see Chapter 3).
**if** $V_{gain} > V_{loss}$ **then**
  do switching according to the calculations
  $S_{ir} \leftarrow S'_{ir}$
**else**
  server configuration remains the same
**end if**
return current configuration

Algorithm 2). This is a best-effort algorithm [54].

This policy works by check whether tiers in any pool is saturated or not. If so the servers are switched from the saturated pools to the other. Switching the servers can not solve the problem of the saturation if the both pools are saturated [54].

**Algorithm 2** The bottleneck-aware switching policy

**Input:** $N_r$, $m_i$, $R$, $K_{ir}$, $S_{ir}$, $v_{ir}$, $\phi_r$ , $t_s$, $t_d$
**Output:** new configuration
**while** bottleneck saturation found in one pool **do**
  **if** found at same tier in the other pool **then**
    return
  **else**
    switch servers to the bottleneck tier
    $m_i \leftarrow m'_i$ and $S_{ir} \leftarrow S'_{ir}$
  **end if**
**end while**
search configurations using Algorithm 3
return current configuration

The bottleneck identification phase works if a bottleneck is detected in either of the pools. If a bottleneck is detected at the same tier in each pool, migrating servers at that

tier will not remove the bottleneck. If a bottleneck exists within a single pool, servers are migrated to remove the bottleneck, subject to revenue improvement.

The local search algorithm (see Algorithm 3) works when there is no bottleneck saturation in either pool. The algorithm uses a nested loop to evaluate server migrations starting from the web tier to the application tier before finally evaluating the database tier. The revenue gain is computed at each stage, with the highest revenue state being chosen.

The input of the algorithm ($N_r$, $m_i$, $R$, $K_{ir}$, $S_{ir}$, $v_{ir}$, $\phi_r$ , $t_s$, $t_d$) are presented in table 2.1. The aim of the proposed algorithm is to find the best configuration that can be obtained from using the nested loop technique. The algorithm starts by calculating the utilisation of the web tier ($U_0^1$ and $U_0^2$), the application tier ($U_1^1$ and $U_1^2$), the data base tier ($U_2^1$ and $U_2^2$) for both pools. The first while loop in the algorithm completes when the utilisation of the web tier in the first pool is greater than the utilisation of the web tier in the second pool. If so, the number of web servers in the second pool is greater than 1, in order to ensure that there are enough servers switched between the pools. After checking that the number of switched servers that reallocated between the pools are more than 1, the number of the web server of the first pool is increased and the number of web server of the second pool is decreased by one. At the same time the service time for both pools are calculated for future process.

The second inner while loop is performed in order to deal with the switching of the servers within the second tier in our multi-tier architecture. As in the first tier (web tier), the comparison between the utilisation of the application tier in both pools are conducted by making sure that the number of switched servers from the source is greater than 1, in order to avoid the situation where no more servers could be reallocated between the two pools. In the situation where the utilisation of the application

tier of the first pool is greater than that of the second pool, and also where the number of application servers in the second pool is more than one, the number of servers in the application tier in the first pool is increased by one and at the same time the number of the application servers is decreased by one. Added to this, the new service time is calculated for both pools for future processing. After dealing with application tier, the algorithm moves to the web tier, as the servers are switched from pool two to pool one where i) the utilisation of the web tier in the first pool is greater than that in the second pool and ii) the number of servers in the second pool is more than one server. In such a situation the new services times are computed for both pools. After that the system revenues are computed using the Equation 3.9 and 3.10 (see Chapter 3 for more details) and the current configuration is stored only when the gained revenue is greater than the lost revenue.

The approach used in this thesis is quite different from that found in [17], where an algorithmic approach is used to optimise a resource allocation problem where resources are given in discrete units; it differs too from the graph-theoretic approach for solving a resource allocation optimisation problem, which is used in [49] and then developed to the case where there are multiple classes of resources in [50].

**Algorithm 3** The configuration search algorithm

```
Input: N_r, m_i, R, K_ir, S_ir, v_ir, φ_r , t_s, t_d
Output: best configuration
Initialisation: compute U_i^1, U_i^2
while U_0^1 > U_0^2 do
  if m_0^2 > 1 then
    m_0^2 ↓, m_0^1 ↑; S_0r^2 ← S_0r^2'
    while U_1^1 > U_1^2 do
      if m_1^2 > 1 then
        m_1^2 ↓, m_1^1 ↑; S_1r^2 ← S_1r^2'
        while U_2^1 > U_2^2 do
          if m_2^2 > 1 then
            m_2^2 ↓, m_2^1 ↑; S_2r^2 ← S_2r^2'; compute V_loss using eq. 3.9 (see Chapter 3).
            S_2r^1 ← S_2r^1'; compute V_gain using eq. 3.10 (see Chapter 3).
            if V_gain > V_loss then
              store current configuration
            end if
            compute new U_i^1, U_i^2
          end if
        end while
        similar steps for U_2^1 < U_2^2
        S_1r^1 ← S_1r^1'; compute new U_i^1, U_i^2
      end if
    end while
    similar steps for U_1^1 < U_1^2
    S_0r^1 ← S_0r^1'; compute new U_i^1, U_i^2
  end if
end while
similar steps for U_0^1 < U_0^2
return best configuration
```

## 2.3 Performance Modelling

As computer systems become more complex, rapidly evolving, and essential to the conduct of business, so understanding the behaviour of these systems in order to provide value for money and performance in terms of throughput become more significant [26]. Intuition and trend extrapolation, experimental evaluation of alternatives, and fi-

nally modelling are the ways that may used in order to understand the behaviour of these computer systems. The first approach is rapid and also flexible, but it is not accurate as it based on using the experience and insight that are difficult to acquire and verify. The second approach which is experimental yields, excellent accuracy, but is not easy to be implemented and is inflexible compared with the first method [26]. Finally, modelling is very flexible compared with theses two approaches, as it is an abstraction of the system where more detail is not needed, and it is less laborious and more flexible than experimentation. The modelling approach is also more methodical, which means that it is a more reliable approach than intuition. Modelling, then, provides a framework for gathering, organising, evaluating, and understanding information about a computer system [26].

### 2.3.1 Queuing Network Models

Queueing network models (QNM) are a tool that is used for system modelling, system performance evaluation and system prediction, including production and manufacturing systems [24] and [26]. In such environments, system resources, users, and transactions are represented by collections of service centres.

A system where multiple customers receive services simultaneously, using multiple service centres and multiple queues and customers may move between the queues, is widely known as queuing network [13]. In such a system, the customer moves from one service centre to the tail of another queue or even to the same service for further transactions. The customer may revisit a queue that has been already visited earlier.

There are some common features in queuing networks including; arrival time, service time and waiting time. The arrival time is the time that the customer arrives at the service centre. The service time and waiting time are defined as the duration during

which the customer was served and the time that had been spent until the customer has been served by the service centre respectively. The services are often provided to the customer as a first-in first-out approach, a last-in first-out approach, a piecemeal approach, or a random order approach [13]. In the first case, the customer at the head of the queue receives the service and any new customer is allocated to the tail of the queue. Where the upcoming customer is served first, the queue is known as stack in such approach. The customer's needs in the third approach are divided between the waiting mode and service mode as the customer job need not to be serviced immediately. In the random approach customers will be served in a random.

■ **Single-class Queuing Networks vs. Multi-class Queueing Networks:**

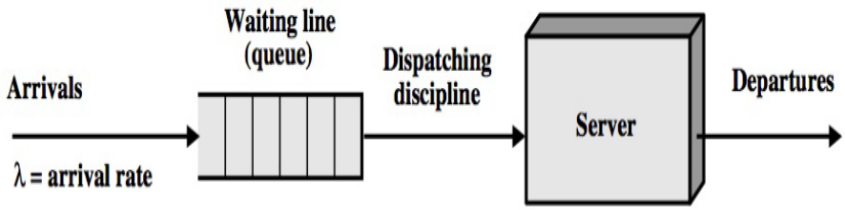


Figure 2.3: A model of a single-server queue [46]

In single-server queues, customers arrive at the service centre and then wait in the queue to be served if required. If the queue is empty the job will be processed immediately. After being served by the service centre, the customer may leave the service centre (see Figure 2.3 [46]). Single server queues can be found every where in the real-world e.g. airports, banks, and public transport.

An attractive design to a queuing network is multi-class queuing network, where more than one service centre is provided. In the situation where all service centres are

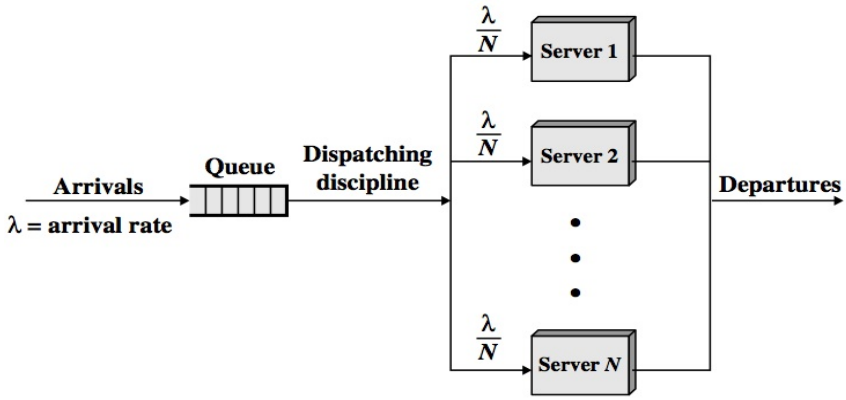


Figure 2.4: A model of a multi-server queue [46]

busy, the customer will be queued until a service centre is free (see Figure 2.4 [46]). Various dispatching techniques can be employed in such situation, e.g. round-robin.

Round-robin is one of the simplest algorithms that can be applied in such environments where a fixed service time is assigned to each process in circular (modulo) order without priority. An example of using the Round-robin algorithm is when the service time given to each job equals 100 milliseconds, and *job1*, for example, takes 250 milliseconds to complete. In such a situation, the round-robin scheduler suspends *job1* after 100 ms and assigns the next job the same period (100 milliseconds). *job1* will be allocated service time only when the other jobs in the queue have themselves been serviced. In other words, *job1* will complete with the following allocations (first allocation = 100 ms, second allocation = 100 ms, and the third allocation = 50 ms) in order to complete the required job.

■ **Open Queueing Networks vs. Closed Queuing Netwrok:**

There are two different approaches to the queueing network model, either an open

model or a closed model. The model of queuing network used is determined based on the limit to the number of requests in the system [40]. That is, if the maximum number of system requests are conserved the queuing network is a closed queuing network. While in an open queuing network, there is no limit to the maximum number of requests present in the system model.

In this thesis we use a multi-class closed queueing network to represent our system of interest. The application is modelled using both /M/1/first-come-first-served and /M/m/first-come-first-served in each station, and it is assumed that servers are clustered at each of the three system tiers.

There are several techniques that can be used in order to solve multi-class closed queuing networks. The first technique used to solve the closed queuing network models is the convolution algorithms [6]. A convolution algorithm works by first computing a set of normalisation constants and then computing the performance measures in terms of these normalisation constants. It relates to this work as it is one of the major algorithms for the solution of closed, product-form queueing networks [25]. However, this thesis uses an alternative technique, Mean Value Analysis (MVA), which is now one of the most popular techniques that can be used to solve closed products-form queuing networks. In MVA the performance measure processes are computed directly without computing the normalisation constant (which improves the efficiency of the technique). The simplicity of MVA (developed firstly in [43]) and the accuracy of its results are the main reasons for using MVA in this thesis (see [9] and [54] respectively). Nowadays MVA is applied in a wide range of applications such as computer systems and networks, financial systems, and medical applications. The MVA in this thesis is solved based on based on Little's law [30] and the Arrival Theorem [45] (see Section 3.3 for more details).

## 2.4 Bottleneck and Admission Control

It is known that the resources that limit the overall performance of the system are the congested ones, referred to as the bottlenecks [8]. A bottleneck in the system may be shifted between tiers according to changes in the workload mix and the number of jobs in the system [3]. It is clear that bottleneck identification should be one of the first steps in any performance study; any system upgrade which does not remove the bottleneck(s) will have no impact on the system performance at high loads [35].

A significant amount of research has been done trying to solve the problem of bottlenecks [3] [15] [27] [8] [22]. The work in [3] [15] [27] studies bottleneck identification for multi-class closed product-form queuing networks for an infinite population, while [8] and [22] study a large population.

The work in [54] uses the convex polytopes approach to identify the bottleneck in two different pools for their chosen configuration using two classes of jobs (gold and silver). The work uses the convex polytopes approach where the set of potential bottlenecks in a network with one thousand servers, two different server pools and fifty customer classes, can be computed in just a few seconds.

The work in [8] presents a technique for identifying the bottlenecks of multi-class queueing networks. This technique is based on the theory of convex polyhedra. The technique assumes that $R = 1, 2, 3, ..., R$ are the set of customer classes and M = 1,2,3,...,M the set of stations in the proposed system. In such a situation $L_{ir} = V_{ir} \times S_{ir}$, where the station, class, average loading, visits to the station, and service requirements are represented by $i, r, L_{ir}, V_{ir}, andS_{ir}$ respectively. This can be collected in the loading matrix $L = L_{ir}$.

$$L = \begin{vmatrix} 20 & 80 \\ 65 & 60 \\ 90 & 30 \\ 25 & 65 \\ 40 & 45 \end{vmatrix}.$$

$L_{21} = 65$ and $L_{22} = 60$ are the loadings imposed on station 2 by class 1 and class 2 customers, respectively. The technique also assumes that the $L_{ir}$ always considers positive quantities, and $L$ is non-singular [8]. Figure 2.5 illustrates the projection $proj(L)$ of matrix $L$, where the projection of station 3 for example can be described as follow; $proj(L3) = (0, 30), (90, 0), (0, 0)$.

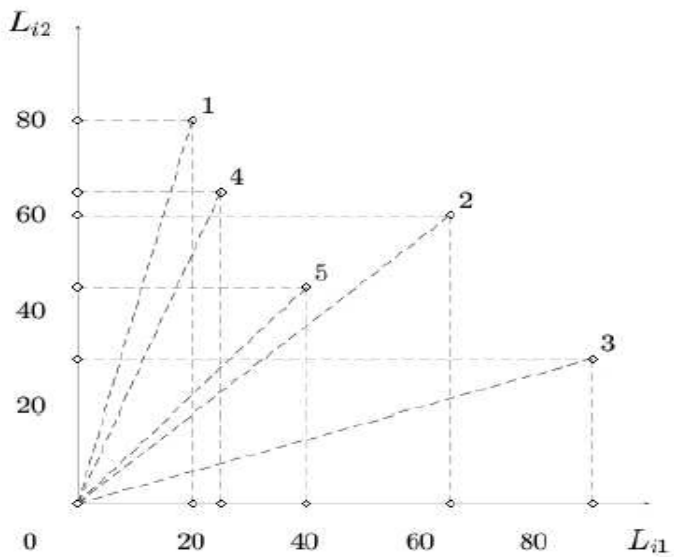


Figure 2.5: Service demands of matrix $L$ and the set of its projections $proj(L)$ in the loadings space [8]

The characteristic polytope of a queueing network with loading matrix $L$ is the convex polytope $T_L = conv(L \cup proj(L))$ [8]. The characteristic polytope of the proposed

matrix is shown in Figure 2.6.

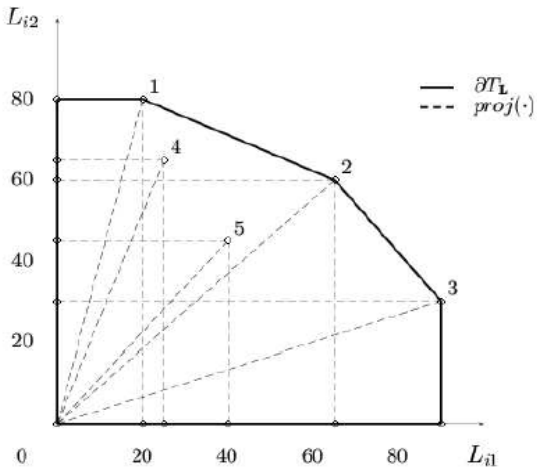


Figure 2.6: Characteristic polytope $T_L$ of the two-class loading matrix $L$ [8]

It has been proved in [8], that all potential bottlenecks are to be found on $\vartheta TL$ (see Figure 2.6).

Figure 2.7 shows example bottleneck identification results using convex polytopes for a chosen server pool configuration. We see that when the percentage of gold class jobs is less than 46.2%, the web server tier is the bottleneck; when it is between 46.2% and 61.5%, the system enters a crossover region, where the bottleneck changes; when the percentage of gold class jobs in pool 1 exceeds 61.5%, the application server tier becomes the bottleneck. Thus it is clear that bottleneck identification should be one of the first steps in any performance study; any system upgrade which does not remove the bottleneck(s) will have no impact on the system performance at high loads, see [35].

Most e-Business applications are subject to enormous variations in workload demand [10]. The traffic to such sites becomes too high e.g. three or four times greater

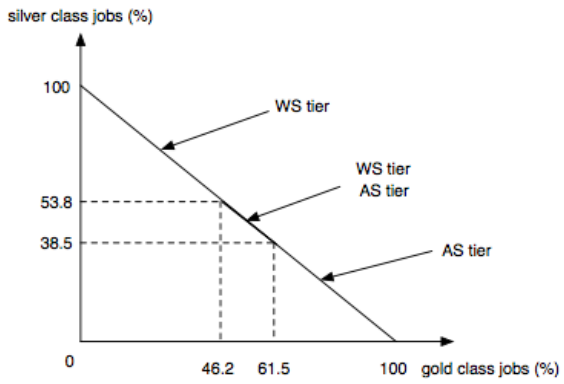


Figure 2.7: Bottleneck identification using convex polytopes [54]

than the average traffic and the server capacity is failed in serving the active customers [41].

System overloading can cause exceptionally long response times for requests or even errors, caused by the timing out of client requests and connections being dropped by the overloaded application. At the same time, the throughput of the system would decrease significantly [12]. A classic example of this was when the normally well-provisioned Amazon.com site suffered a forty minute downtime due to an overload during the popular holiday season in November 2000 (see [51]). Another example was the failure of the CNN.com website after the terrorist attacks on the United States on September 11, 2001 [16]. As a result of these (and other) case studies, and subsequent research, admission control is applied in order to deal with system overloading.

The work in [41] has developed resource management schemes in order to improve the revenue throughput of Internet application. Resource management policies for e-commerce sites as stated in [41] should be based on the behaviour of active customers (e.g. as are they navigate through the site, or going from browsing to searching, or selecting items, or adding them to their shopping carts and paying). In other words,

the resources should be reassigned based on the importance of the customer, that is the customer who is about to buy from the sites should be given more priority than the customer who is still browsing through the sites. Added to this, customers' priorities change dynamically based on the following aspects; i) the customer profiles, ii) the length of the current session length, iii) the amount of money accumulated in the customers shopping cart, and iv) the states visited in the customer behaviour model graph (CBMG) using three priority classes – high, medium, and low – where the customers transition between these three priority classes based on the previous proposed aspects.

A simple admission control policy has been developed by [54]. It works by dropping less valuable requests when the response time exceeds a threshold, and therefore maintaining the number of concurrent jobs in the system at an appropriate level. This policy has been applied in this thesis in order to maintain the number of requests access to the proposed system.

The scheme that used in this thesis is based on assigning priorities to requests and ensuring that less important requests are rejected when the system is overloaded .

## 2.5 Fundamental Laws

The system in this thesis is modelled using a multi-class closed queuing network to compute the various performance metrics. The advantage of using an analytical model is that we can easily capture the different performance metrics, and identify potential bottlenecks without running the actual system. The model can also react to parameter changes when the application is running (e.g. from the monitoring tools or system logs) and make dynamic server switching decisions to optimise pre-defined performance metrics [54]. The different mathematical rules that are used in this thesis will be presented in this section, including: Utilization Law, Force Flow Law, Service De-

mand Law, Little's Law, and Response Time Law.

In these definitions:

$T$: is the length of time the system is monitored;

$B$: is the length of time that the resource is busy;

$C$: is the total number of completed requests.

### 2.5.1 Utilization Law

The utilization $U$ in a single resource $i$ is defined as the fraction of the time that the resource is busy ($U_i = B_i/T$); the throughput $X$ of the resource $i$ can be obtained after calculating the total number of completed jobs $C$ during a specific period ($X_i = C/T$). The average service time $S$ is computed by dividing the length of time that the resource is busy ($B$) and the number of requests that are already completed by the selected resource ($C$). After computing the previous three parameters (the throughput, the utilization, and the service time), the first fundamental law, which is known as the *UtilizationLaw*, can be computed by multiplying the throughput $X_i$ of the resource $i$ and the average service time at that resource.

$$U_i = X_i \times S_i \tag{2.1}$$

As an example, suppose that a network segment transmits 100 packets each second and the transmission time of each of these packets is equal to 1.5 milliseconds. In such a situation and by using the utilization law, the utilization of that segment is 15% and it is computed as follows; $100 \times 0.0015 = 0.15$.

### 2.5.2 Forced Flow Law

The $ForceFlowLaw$ focuses on the average number of visits by a request to the resource $i$, which is known as $V_i$, and the average number of requests that have been completed within a given period $X_0$. Based on these two parameters the throughput $X_i$ of the resource $i$ can be obtained.

$$X_i = V_i \times X_0 \tag{2.2}$$

Suppose that there 3,600 transactions were executed within half an hour and each of these transaction perform 2 I/O operations on average on the database server.

The utilization of the disk can be calculated based on the Force Flow Law and the Utilization Law respectively. First of all the Force Flow Law is applied in order to compute the total throughput, $X_i = V_i \times X_0$, where ($X_0 = 3600/1800 = 2tps$) and $V_i$ is given (it is equal to 2). Therefore by using the Force Flow Law, the throughput ($X_i = 2 \times 2 = 4tps$). After that the Utilization Law can be used to calculated the utilization of the database disk as follows; $U_i = X_i \times S_i = 4 \times 0.02 = 0.08 = 8\%$, where each disk I/O takes on average 20 milliseconds to be executed.

### 2.5.3 Service Demand Law

The third fundamental law is the Service Demand Law $D_i$, which is defined as the total time that the request spends in service at resource $i$. It is can be computed by using the system throughput $V_i$ and the utilization $U_i$. The Service Demand Law is related to the Force Flow Law and Utilization Law.

$$D_i = V_i \times S_i = (X_i/X_0) \times (U_i/X_i) = U_i/X_0 \tag{2.3}$$

The Service Demand Law can be applied to compute the service demand of the data base disk that is used in the previous example, as $D_i = (U_i/X_0) = (0.08/2) = 0.04$ sec.

### 2.5.4 Little's Law

Little's Law focuses on the relationship between the average time spent in the system $R$ and the throughput of that system $X$, which produces the average number of requests in the system $N$.

$$N = X \times R \tag{2.4}$$

By applying Little's Law, the average response time $R$ can be calculated for the server that was monitored. For example, suppose that in 1 hour where there were 14,400 I/O operations executed during the monitoring period. As the throughput of the server $X$ is equal to 4 request per second (14400 / 3600 = 4), Little's Law can be applied in order to compute the response time of the servers where the average number of active requests was found to be 1 request; $R = N/X = 1/4 = 0.25$.

### 2.5.5 Response Time Law

The simple relationship between the parameters – the average response time ($R$), the number of requests from the sources ($M$), the throughput of the requests processing ($X_0$), and the average time elapsed between the reply from the request and the submission of the new requests by the same source ($Z$) – is called the Response Time Law.

In order to compute the average number of requests being processed by the system, Little's Law is used ($N = R \times X_0$). The second step is to compute the total number of

requests in the system ($M$), by adding the average number of requests being processed ($N$) and the average number of requests in the (think state) $\overline{M}$, ($M = N + \overline{M}$). Based on this and also Little's Law, $M = N + \overline{M} = (R \times X_0) + (Z \times X_0) = (R + Z) \times X_0$, the Response Time Law can be obtained using the following equations;

$$R = (M/X_0) - (Z). \tag{2.5}$$

An example of using the Response Time Law is when the average think time of a web service needs to be computed. Consider a web service receives on average 7,200 requests each hour, from 200 different customers, and the average response time was measured as 2.5 seconds. In such situations and in order to calculate the average think time ($Z$), the Response Time Law can be applied here to compute such a parameter as follows; ($Z = M/X_0 - R = (200/(7200/3600)) - 2.5 = (200/2) - 2.5 = 97.5 seconds$). This means that the average time since any response to a reply is received and a new request is submitted by the customer is equal to 97.5 seconds.

Table 2.2 summarises all the fundamental performance laws relevant to this thesis.

Table 2.2: Fundamental Performance Laws

| Law | Equation |
|---|---|
| Utilization Law | $U_i = X_i \times S_i$ |
| Forced Flow Law | $X_i = V_i \times X_0$ |
| Service Demand Law | $D_i = V_i \times S_i = U_i/X_0$ |
| Little's Law | $N = X \times R$ |
| Response Time Law | $R = (M/X_0) - (Z)$ |

## 2.6 Solving Multi-Class Closed Queueing Networks

Queuing networks can be used to model typical multi-tiered enterprise systems. In such situations, multi-class queuing networks are the ideal solution. A closed queuing network is applied in order to represent the enterprise system, as there is a limit to the number of simultaneous customers logged into the enterprise system at any one time [39]. The ability to compute performance metrics, identify potential bottlenecks and, importantly, investigate a wide variety of hypothetical scenarios, without running the actual system are considerable benefits of applying the analytical models. As a result, in this thesis the applications that have been modelling using these analytical models. One should thus envisage such a model running alongside a real system, where the model can react to parameter changes as the application is running (e.g. from monitoring tools or system logs) and making dynamic configuration decisions to optimise pre-defined performance metrics [54].

We use a multi-class closed queueing network to represent our system of interest, where C, WS, AS, and DS represent the Client, Web Server, Application Server, and Database Server respectively. The application is modelled using both -/M/1 first-come-first-served and -/M/m- first-come-first-served in each station, and it is assumed that servers are clustered at each tier.

As stated previous, there are several approches that can be used in order to solve multi-class closed queuing networks including; convolution algorithms (CA) [6], and mean value analysis (MVA) [5].

### 2.6.1 Mean Value Analysis

The traditional technique that used in order to solve the queuing networks (QN) was developed to formulate a system of algebraic equations for the joint probability distribution of the vector-valued system state by normalizing the product terms to form a proper probability distribution [43]. This process which is based on the normalisation to product terms (e.g. mean queue sizes, meaning waiting times, utilizations, and throughputs) has been proved to be computationally limited in the case of networks with closed routing chains. The mean value analysis (MVA) (it was developed by [43]) on the other hand is defined as a tool used to measure the mean performance measures in closed queueing models is based on the relation between the mean waiting time and the mean queue size of a system with one customer less. This mean-value equations, augmented by Little's equation can be easily solved numerically with no need to compute normalization constants.

The developed algorithms (MVA) which is used in this work is simple and avoid overflow/underflow problems which may arise with the other algorithms [43]. Added to this the MVA is considerably faster in the case of multi-servers. The accuracy of the MVA results [54] and its simplicity compared with convolution algorithms [9] plus the previous reasons are the main reasons for using MVA in this thesis.

Nowadays MVA is applied in a wide range of applications such as computer systems, computer networks, financial systems, and medical applications. The MVA in this thesis is solved based on based on Little's law [30] and the Arrival Theorem [45]. MVA computes the model's performance of the queuing network model based on recursively using three different equations including; the residence time equation, the throughput equation, and the queue length equation [40].

It is possible to solve the queuing networks using Approximation Mean Value Analysis (AMVA), which can solve the queueing network faster than MVA. The accuracy of such algorithm however has lots of concerns. Because of this, the MVA approach has been applied in this thesis to solve the multi-class closed queuing networks.

The description of the system model that applied in this thesis based on the mean value analysis (MVA) is explained in details including its equations in Chapter 3 in section 3.3.1.

Additional background research is discussed at the beginning of each of the following subsequent chapters. The thesis is structured in this way to ensure that the appropriate background research is presented in the context of of the four research contributions that this thesis makes (see Sections 3.2, 4.2, 5.2, and 6.2)

# Chapter 3

# Impact of Server Allocation Time on Dynamic Server Switching

## 3.1 Introduction

Internet hosting centres are often used for the cost-effective hosting of enterprise applications. Typically, these enterprise applications employ a multi-tier architecture, which provides a clear separation of roles between the tiers. Commonly a multi-tier architecture consists of three tiers; a client-facing web tier, an application tier for the application logic and a data persistence tier that is usually comprised of a relational database management system (RDBMS). At each tier servers may be clustered to provide high-availability and improve performance. An Internet hosting centre may host many multi-tier applications for its clients, each of which will have a separate service level agreement (SLA).

Workloads for internet services have been shown to be bursty with large variations in demand [1], [4], and [57]. When pre-defined resource allocation policies are in use they may not be able to handle large surges in traffic, leading to SLA violations and reduced revenues. Dynamic resource allocation systems have been shown to provide a significant increase in revenue in such environments by reallocating servers into a

more profitable configuration [54].

There are several considerations in a dynamic resource allocation system including the decision interval, which is the time taken between evaluations of the policy, and the server reallocation time, which is the time taken to reallocate servers. In this chapter the impact of the time taken to reallocate servers between applications (switching time) on the system's performance is considered alongside the different switching servers policies.

### 3.1.1 Chapter Contributions

The specific contributions of this chapter can be summarised as follows:

- Evaluating the effects of switching time on dynamic resource allocation in order to discover mechanisms for selecting system configurations which best match the switching interval;
- Examining the impact of the combination of switching policies and switching time on the total performance of the system used in this work;
- Studying the impact of applying the admission control technique (see chapter 2 section 2.4 for more details about the admission control technique) on the developed system and to compare its performance with that obtained from applying no admission control.
- The results are based on five different workloads, three of them are obtained from real-world workload traces from several sources, including from the San Diego Supercomputer Centre, from the ClarkNet Internet access provider for the Metro Baltimore-Washington DC area and, from the NASA Kennedy Space

Center web-server in Florida, and the rest of the workloads are generated from two synthetic workloads in order to ensure consistent and improved results.

### 3.1.2 Chapter Structure

This chapter is structured as follows; Section 3.2 reviews the additional related work for this chapter, Section 3.3 describes the model of the system and the revenue function used in order to compute the system performance. Sections 3.4 and 3.5 present the description of the experimental setup and results respectively. Finally Section 3.6 summarises the chapter.

## 3.2 Additional Related Work

Revenue maximisation is a key goal of many dynamic resource allocation systems. In [41] the authors use priority queues to offer differentiated services to different classes of request to optimise company revenue. Different priorities are assigned to differed requests based upon their contributions to the revenue.

The work in [31] focussed on maximising profits of best-effort requests when combined with requests requiring a specific quality of service (QoS) in a web farm. In [31] it is assumed that arrival rates of requests are static, whilst the arrival rates in our work are dynamic. The authors attempt in [21] to maximise revenue by partitioning servers into logical pools and switching servers at runtime. We consider switching in a multi-tier (3 tiers) environment, where the work in [21] just consider the switching on the first tier (Web Server).

Chapter 2 presents the two server switching policies used here; the proportional switching policy (PSP) and the bottleneck-aware switching policy (BSP). This chapter examines the impact of the server reallocation time on the revenue achieved by the

system.

A number of researchers have studied bottleneck identification (e.g. [29]) for multi-class closed product-form queueing networks where there is no limit to growth. We employ the work found in [8] and [54] (as part of our collaboration with HP Labs, IBM and the UK National Business to Business Centre), where convex polytopes are used for bottleneck identification.

Admission control is applied in order to deal with system overloading. This scheme is based on assigning priorities to requests and ensuring that less important requests are rejected when the system is overloaded (see chapter 2).

## 3.3 Modelling of Multi-tiered Internet Services and Revenue Functions

### 3.3.1 The System Model

A multi-tiered Internet service can be modelled using a multi-class closed queuing network [57][44]. The closed queuing network model used here is illustrated in Figure 3.1, as commonly found, a multi-tier architecture consists of three servers (tiers); a client-facing web server (WS), which is responsible for receiving the requests from the client (C) and sending the response back, an application server (AS) used for the application logic, and a data-persistence tier that is usually comprised of a relational database management system (DS).

In a multi-class closed queuing network $S_{ir}$ represents the service time which is defined as the average time spent by a class-$r$ job during a single visit to station $i$ and $v_{ir}$ symbolizes the visiting ratio of class-$r$ jobs to station $i$ (the notation used in the system model is summarised in Table 4.1).

Service demand $D_{ir}$ is defined as the sum of the service times at a resource over

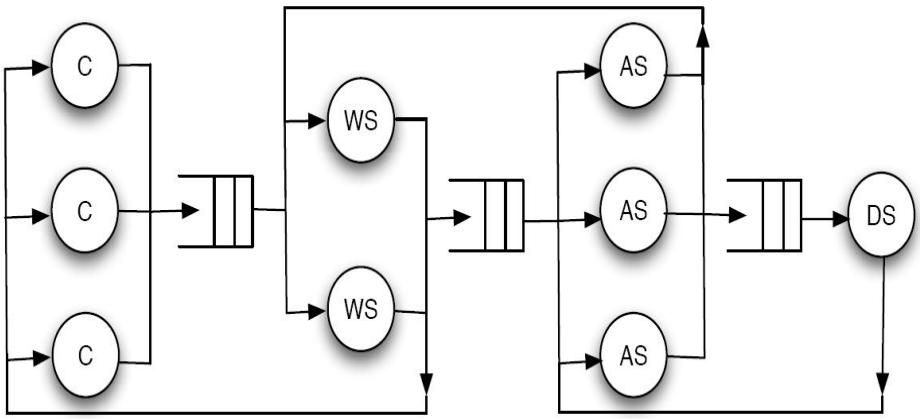


Figure 3.1: A model of a typical configuration of a cluster based multi-tiered Internet service. C represents customer machines; WS, AS and DS represent web servers, application servers and database servers, respectively.

all visits to that resource during the execution of a transaction or request ($D_{ir} = S_{ir} \cdot v_{ir}$) [29]. The total population of the network ($K$) is defined as the total of the total population of customers of class $r$ ($K_r$):

$$K = \sum_{r=1}^{R} K_r \tag{3.1}$$

In modern enterprise systems, servers are often clustered together so both -/M/1-FCFS and -/M/m-FCFS in each station should be measured as a consequence of using a cluster of servers in each tier in our model. The response time of a class-$r$ job at station $i$ can be computed as follows [5],

$$\overline{T}_{ir}(k) = \begin{cases} D_{ir}\left[1 + \sum_{r=1}^{R} \overline{K}_{ir}(k - 1_r)\right], & m_i = 1 \\ \dfrac{D_{ir}}{m_i}\left[1 + \sum_{r=1}^{R} \overline{K}_{ir}(k - 1_r) \right. \\ \left. + \sum_{j=0}^{m_i-2} (m_i - j - 1)\pi_i(j \mid k - 1_r)\right], & m_i > 1 \end{cases} \tag{3.2}$$

Where,

Table 3.1: Notation used in the system model

| Symbol | Description |
|---|---|
| $S_{ir}$ | Service time of job class-$r$ at station $i$ |
| $v_{ir}$ | Visiting ratio of job class-$r$ at station $i$ |
| $N$ | Number of service stations in QN |
| $K$ | Number of jobs in QN |
| $R$ | Number of job classes in QN |
| $K_{ir}$ | Number of class-$r$ job at station $i$ |
| $m_i$ | Number of servers at station $i$ |
| $\phi_r$ | Revenue of each class-$r$ job |
| $\pi_i$ | Marginal probability at centre $i$ |
| $T$ | System response time |
| $D_r$ | Deadline for class-$r$ jobs |
| $E_r$ | Exit time for class-$r$ jobs |
| $P_r$ | Probability that class-$r$ job stays |
| $X_r$ | Class-$r$ throughput before switching |
| $X_r^{'}$ | Class-$r$ throughput after switching |
| $U_i$ | Utilisation at station $i$ |
| $t_s$ | Server switching time |
| $t_d$ | Switching decision interval time |

- there are $k$ jobs in the queuing network, for $i = 1, \ldots, N$ and $r = 1, \ldots, R$.
- $(k - 1_r) = (k_1, \ldots, k_r - 1, \ldots, K_R)$ is the population vector with one class-$r$ job less in the system.

The total system response time for the system $T_i(k)$ is the sum of response time for each tier:

$$T_i(k) = \sum_{r=1}^{R} T_{ir}(k) \tag{3.3}$$

For the case of multi-server nodes ($m_i > 1$), it is necessary to compute the marginal probabilities. The marginal probability that there are $j$ jobs ($j = 1, \ldots, (m_i - 1)$) at the

station $i$, given that the network is in state $k$, is given by [5],

$$\pi_i(j \mid k) = \frac{1}{j}\left[\sum_{r=1}^{R} \frac{v_{ir}}{S_{ir}} X_r(k)\, \pi_i(j-1 \mid k-1_r)\right] \tag{3.4}$$

The throughput of class-$r$ jobs can be calculated using Little's law [29] by dividing the total population of customers of class-$r$ $K_r$ by the sum of the visiting ratio $v_{ir}$ multiplied by the sum of the mean response time of each tier,

$$X_r(k) = \frac{k_r}{\sum_{i=1}^{N} v_{ir} \overline{T}_{ir}(k)} \tag{3.5}$$

By applying the Little's Law again with the Force Flow Law [29], the mean queue length $\overline{K}_{ir}$ is obtained by multiplying the throughput $X_r(k)$, the mean response time $T_{ir}(k)$, and the visiting ratio $v_{ir}$.

$$\overline{K}_{ir}(k) = X_r(k) \cdot \overline{T}_{ir}(k) \cdot v_{ir} \tag{3.6}$$

Where, $\overline{K}_{ir}(0, 0 \ldots, 0) = 0$, $\pi_i(0 — 0) = 1$, and $\pi_i(j — 0) = 0$. Then the system response time, throughput and mean queue length in each tier can be calculated after $K$ iterations.

In multi-class product form queuing networks, per-class station utilisation can be computed using the following equation [32],

$$U_{ir}(k) = \frac{k_r D_{ir}}{\sum_i D_{ir}[1 + \overline{K}_i(k - 1_r)]} \tag{3.7}$$

The total station utilisation $U_i(k)$ is the sum of per-class station utilisation, ($U_i(k) = \sum_{r=1}^{R} U_{ir}(k)$).

### 3.3.2 Modelling the Revenue Function

A session is defined as a sequence of requests of different types made by a single customer during a single visit to a site [37]. An Internet hosting centre supports many

multi-tier applications for its clients, each of which will have separate service level agreements (SLAs) as introduced in Chapter 2. The SLA defines the level of service agreed between the client and the hosting centre and may include performance and availability targets, with penalties to be paid if such targets are not met. It is in the interests of the service hosting centre to ensure that its SLAs are met so that it can maximise its revenue, whilst ensuring that its resources are well utilised. In other words, the maximum revenue is obtained when a client request is met within the deadline, while revenue obtained from requests which are not served within the deadline decrease linearly to zero, at which point the request exits the system.

Equation 3.8 explains how the probability function of the request execution in the system (which is donated by $P(T_r)$) works in our model where $r$, $D_r$, $T_r$, and $E_r$ represent the request and its deadline, response time, and dropped time from the system respectively.

$$P(T_r) = \begin{cases} 1, & T_r < D_r \\ \dfrac{T_r - D_r}{E_r - D_r}, & D_r \leq T_r \leq E_r \\ 0, & T_r > E_r \end{cases} \tag{3.8}$$

The first part of Equation 3.8 states that the full revenue will be contributed by the request if it is processed before the deadline $D_r$. It is clear from the second part of the equation that the gained revenue by the request is calculated by dividing the difference between the request response time $T_r$ and its deadline $D_r$ by the difference between request dropped time from the system $E_r$ and its deadline $D_r$. The request gains no revenue when its response time $T_r$ is greater than the time at which the request exits the system $E_r$.

With respect to the probability of the request execution, the lost $V^i_{loss}$ and gained $V^i_{gain}$ revenue are calculated with the assumption that the servers are switched from

pool $i$ to pool $j$ using the Equations 3.9 and 3.10 respectively [54].

Note that because the servers are being switched, they can not be used by both pools $i$ and $j$ during the switching process and the time that the migration takes cannot be neglected. The revenue gain from the switching process is calculated during the switching decision interval time $t_d^{'}$ as shown in Equation 3.10 where the switching decision interval time is greater than the switching time.

$$V_{loss}^{i} = \sum_{r=1}^{R} X_r^i(k^i)\phi_r^i P(T_r)t_d - \sum_{r=1}^{R} X_r^{i'}(k^i)\phi_r^i P(T_r)t_d \tag{3.9}$$

$$V_{gain}^{j} = \sum_{r=1}^{R} X_r^{j'}(k^j)\phi_r^j P(T_r)(t_d - t_s) - \sum_{r=1}^{R} X_r^j(k^j)\phi_r^j P(T_r)(t_d - t_s) \tag{3.10}$$

After calculating the achieved and lost revenue by using Equations 3.9 and 3.10, servers may be switched between the pools. In our model servers are only switched between the same tiers, and only when the revenue gain is greater than the revenue lost.

## 3.4 Experimental Setup and The Workload

### 3.4.1 Experimental Setup

In the experiments in this chapter two applications are modelled as running on two logical pools. Each of these is multi-tiered, with each tier comprising a cluster of servers. The service time $S_{ir}$, the visiting ratio $v_{ir}$ and the remaining experimental parameters are based on realistic (i.e. sampled) values, or from those supplied in supporting literature [54]. Table 3.2 summarises the main experimental parameters which are used in this chapter.

The focus of the experimentation is to investigate how the time taken to reallocate

Table 3.2: The main experimental parameters.

| | | Pool 1 | | Pool 2 | |
|---|---|---|---|---|---|
| | | Silver | Gold | Gold | Silver |
| Service time (sec) | WS | 0.07 | 0.1 | 0.05 | 0.025 |
| | AS | 0.03125 | 0.1125 | 0.01 | 0.06 |
| | DS | 0.05 | 0.025 | 0.0375 | 0.025 |
| Visiting ratio | WS | 1.0 | 0.6 | 1.0 | 0.8 |
| | AS | 1.6 | 0.8 | 2.0 | 1.0 |
| | DS | 1.2 | 0.8 | 1.6 | 1.6 |
| Deadline (sec) | | 20 | 15 | 6 | 8 |
| Exit point (sec) | | 30 | 20 | 10 | 12 |
| Revenue unit | | 2 | 10 | 20 | 4 |
| Number of servers | WS | 4 | | 5 | |
| | AS | 10 | | 15 | |
| | DS | 2 | | 3 | |

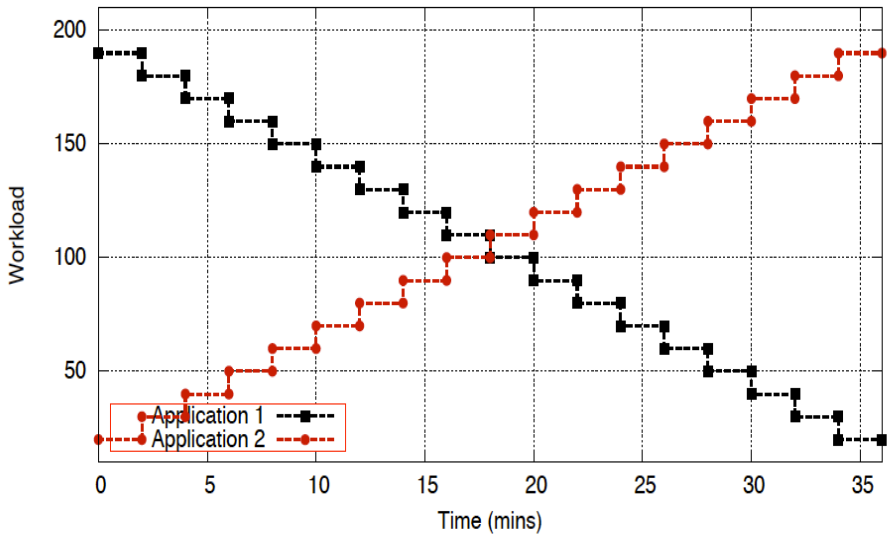


Figure 3.2: The First Inversely Proportional Workload

servers affects the revenue derived from the system. We have fixed the decision interval for the policies at 60 seconds, and experimented with reallocation times of 5 to 55 seconds.

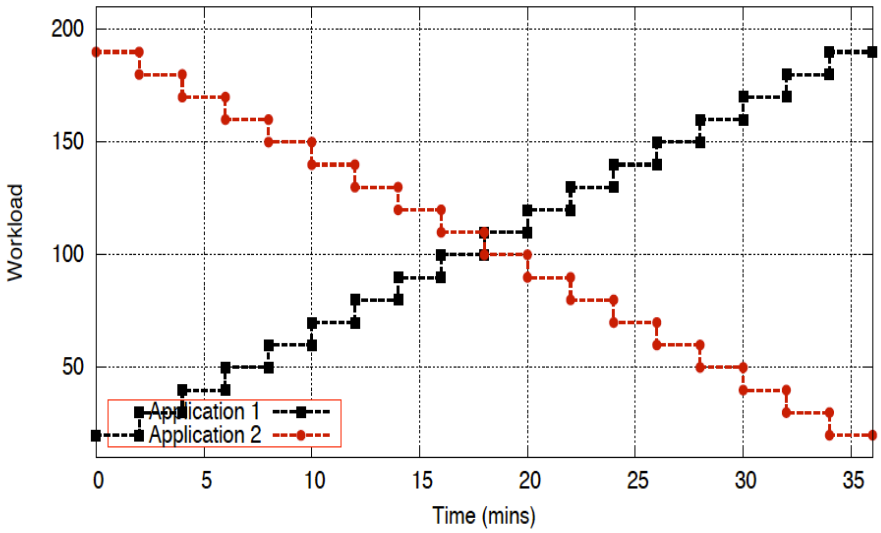


Figure 3.3: The Second Inversely Proportional Workload

### 3.4.2 The Workload

The workload is defined as the set of all inputs the system receives from its environment during any given period of time [40]. In this study we use two classes of workload: synthetic and real-world.

- Synthetic - It is not uncommon in studies such as these to use synthetic workloads. This is done because it allows system architects (such as ourselves) to observe the resulting system behaviour as a result of one specific workload characteristic (as opposed to a real-world work stream that contains many different characteristics). Here we use two inversely proportional workloads, Figures 3.2 and 3.3, whose workloads switch over time. We use the synthetic workload to illustrate the properties of our system with respect to this single workload characteristic. Clearly, this captures only one case, and can not be assumed to represent

sufficient testing, thus we also run the experiments on additional real-world test cases. The synthetic workload is however useful for observing cause and effect, and for tuning the sensitivity of the system before it is deployed for real.

- Real-world - Three real-world Internet traces are also used. These contain two days, two weeks, and two months worth of HTTP requests, where the first one is generated from two real-world Internet traces containing 76,086 requests in total and each of which contains a days worth of HTTP requests to the EPA WWW server located at Research Triangle Park, NC and the SDSC WWW server located at the San Diego Supercomputer Center in California respectively [27], and the second real-world Internet traces has been collected from ClarkNet WWW server which is a full Internet access provider for the Metro Baltimore-Washington DC area [2]. This workload contains 3,328,587 requests issued to the server during the period of two weeks, while the third real-world workload used in this research is obtained from the NASA Kennedy Space Center web-server in Florida [2]. This trace contain 3,461,612 requests spanning two months.

In typical fashion (see also [40]) we characterise this workload to form a workload model, which can then be used as the input to our system model. We consider a typical Web farm as in [52], which records the hypertext transfer protocol (HTTP) service requests and aggregates them over small time intervals of length $\Delta > 0$ to obtain a time series. In this work the period of 5 minutes has been chosen to represent the interval time over one day, two weeks, and one month periods giving a total of 288 intervals in a day (see example in [52]).

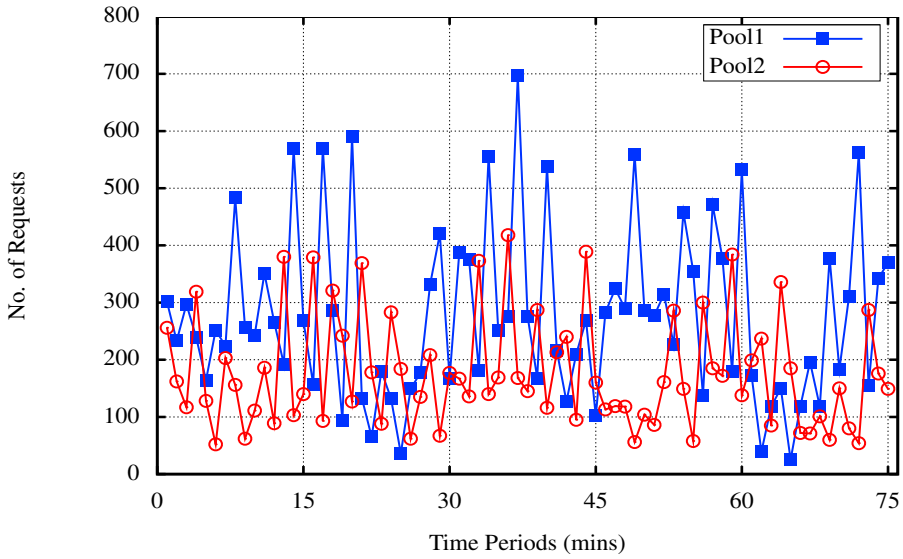


Figure 3.4: A sample of the total requests in the real-world workload for both application pools

A sample of the real-world Internet traces (the traces that obtained from the NASA Kennedy Space Center web-server in Florida) for the two provided applications is shown in Figure 3.4.

## 3.5 Experimental Results

Experiments are conducted over five different workloads including; two inversely proportional workloads and three Internet Traces its results are described in the following sections.

In the presentation of results we considered adding a statistical assessment of whether the observed results reflect a pattern of behaviour or are simply a result of chance. That is, are the results simply observed or a result of experimental error, or are they statistically significant. While this might seem a sensible course of analysis, we note that previous research in this area has not adopted this approach (see [18], [19],

[40], and [54]); we discuss here why this is the case, and therefore justify our own reasons for not introducing tests of significance.

Consider two server pools $A$ and $B$, who themselves are responsible for generating revenue $A_r$ and $B_r$ (the system revenue is simply $A_r+B_r$). Let us assume (for simplicity) that work enters the system via two work streams $WS_a$ and $WS_b$, the first feeding server pool $A$, the second feeding server pool $B$. If a job arrives in $WS_a$, and there is server capacity in $A$ to service that request, then the job is passed to a free server in $A$; if there is no server capacity in A then the job is queued. $WS_b$ operates in a similar fashion.

Server pools $A$ and $B$ will work to capacity, and the job queue to each will depend on that capacity, the job arrival rate and also the job distribution. Each of these three criteria impact the revenue generated ($A_r$ and $B_r$). Thus if the capacity in each server pool is fixed, and we assume each job is of equal size, system revenue will depend on the job arrival rate and job distribution in $WS_a$ and $WS_b$. As observed therefore, the revenue resulting from a system of this type will be dependent on the workload. The reader will recognise that a detailed exploration of this example is governed by the mathematical properties of queueing theory [47], and that while this has limitations (e.g. infinite queue capacity, infinite customers), these models hold up in reality as the wealth of literature on this topic demonstrates.

Switching servers between pools $A$ and $B$ clearly impacts $A_r$ and $B_r$, as the capacity of each server pool changes. Thus the system revenue may also change as servers are switched, but this is again dependent on the properties of $WS_a$ and $WS_b$. Given that $WS_a$, $WS_b$ and the server capacity in $A$ and $B$ are deterministic, any results are reproducible (both mathematically and in practice). We therefore choose to document the percentage gain (or loss) of system revenue as different server switching schemes are employed over different workloads (as is the case with previous literature in this

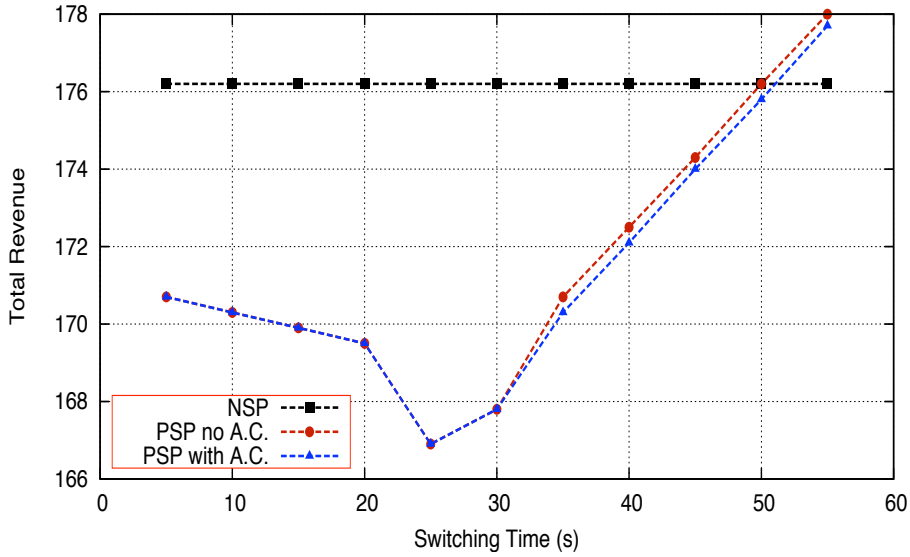


Figure 3.5: Revenue Generated by the Proportional Switching Policy (PSP) Over Workload One at Different Reallocation Times

area).

### 3.5.1 Experiment One

The revenue generation from applying the two switching policies – PSP and BSP – over the first inversely proportional workload are shown in Figures 3.5 and 3.6 respectively.

Figure 3.5 shows the results of the proportional switching policy over the first workload. The policy demonstrates a decrease in revenue, as the reallocation time increases from 5 to 20 seconds. After that there is a significant reduction in revenue, which then increases dramatically as the reallocation time increases. The behaviour of the policy changes throughout the experiment, with the policy migrating many servers when the reallocation time is small and fewer as the time increases.

The proportional switching policy does not demonstrate an improvement in revenue over the pre-defined allocation at all reallocation durations, with just one excep-

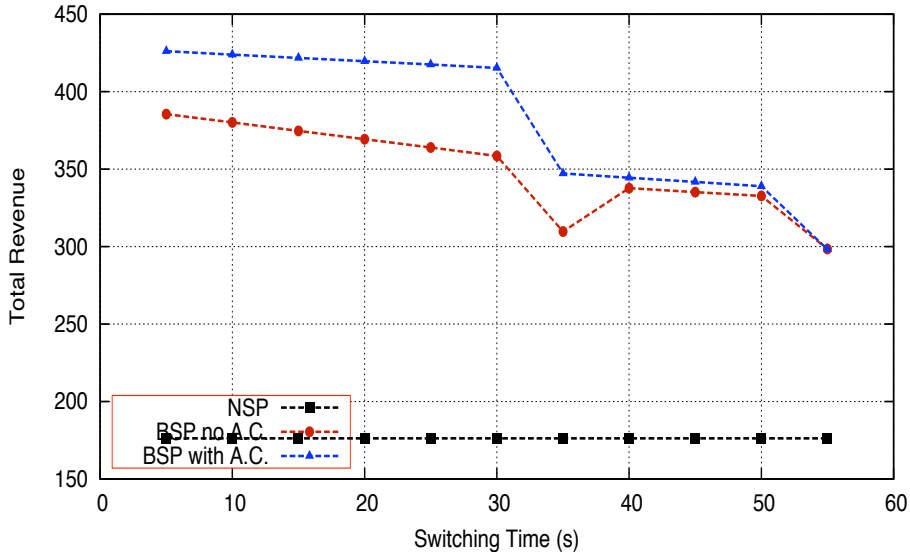


Figure 3.6: Revenue Generated by the Bottleneck Aware Switching Policy (BSP) Over Workload One at Different Reallocation Times

tion when the switching time equal to 55. The use of admission control has no effect on the revenue generated by the policy.

The number of switches servers begins with 6 switches when the switching time is 5 second and remain the same when the switching time changes to be 10, 15, and 20 seconds. After that the number of switches servers dropped up to just 2 switches when the switches time are 25 and 30 seconds. It increase from two switches to be 3 switches when the switching time are 35, 40, 45, 50, and 55 seconds. This behaviour of the switches servers happens when the PSP is applied with and without the admission control (AC).

The bottleneck-aware switching policy (BSP) is the best performing policy over workload one (see Figure 3.6). It provides a significant improvement in the revenue generated by the system. The improvement decreases linearly when the switching times are from 5 seconds to 30 seconds, where there is a drop at 35 seconds, before the

revenue decreases linearly again. The large drop in revenue at 35 seconds is caused by an increase in the number of server migrations from 12 at a 30 second duration to 15 at a 35 second duration.

Using the admission control (AC) policy enhances the revenue generated at all reallocation times, however the enhancement is reduced when the reallocation time increases beyond 30 seconds.

The number of switches servers when the BSP is applied along side AC and without AC rises with the increase of the switching time. It increases from 8 and 12 switches when the switching time is 5 to reach a 13 switches for both scenario with and without AC.

### 3.5.2 Experiment Two

Figures 3.7 and 3.8 show the result obtained from applying the two switching policies – PSP and BSP – over the second workload respectively.

The PSP (see Figure 3.7) performs better than a pre-defined allocation at all intervals over the second workload. In this scenario the use of admission control has no impact on the revenue obtained through the use of the PSP policy.

The results from applying the second switching policy (BSP) over the second workload are shown in Figure 3.8. This policy most clearly shows a linear relationship between the reallocation time and the revenue generated. The linear relationship is preserved with or without the use of the admission control policy. The policy demonstrated significant improvements in revenue over a pre-defined allocated system. Applying the AC policy generates less revenue comparing the that from applying no AC. It has been outlined how aggressive admission control negatively affects a system over light load (see [54]).

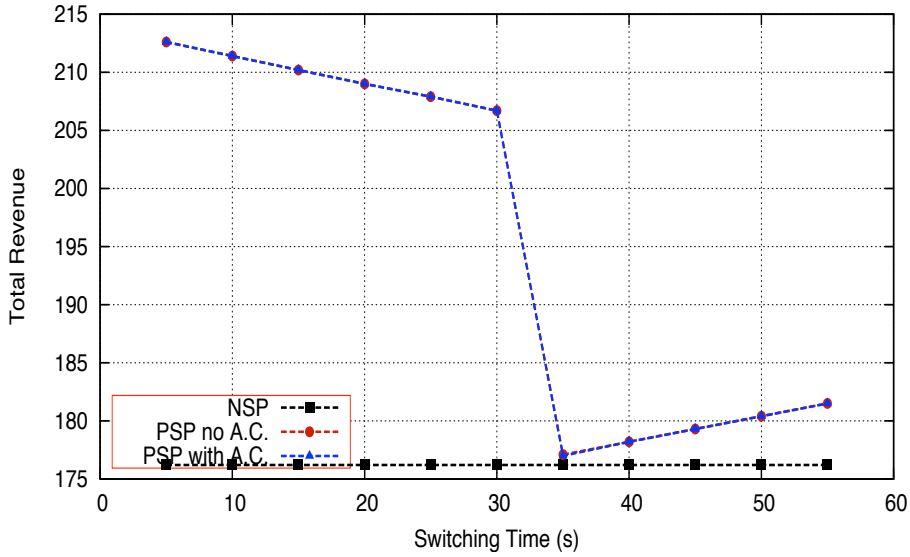


Figure 3.7: Revenue Generated by the Proportional Switching Policy (PSP) Over Workload Two at Different Reallocation Times

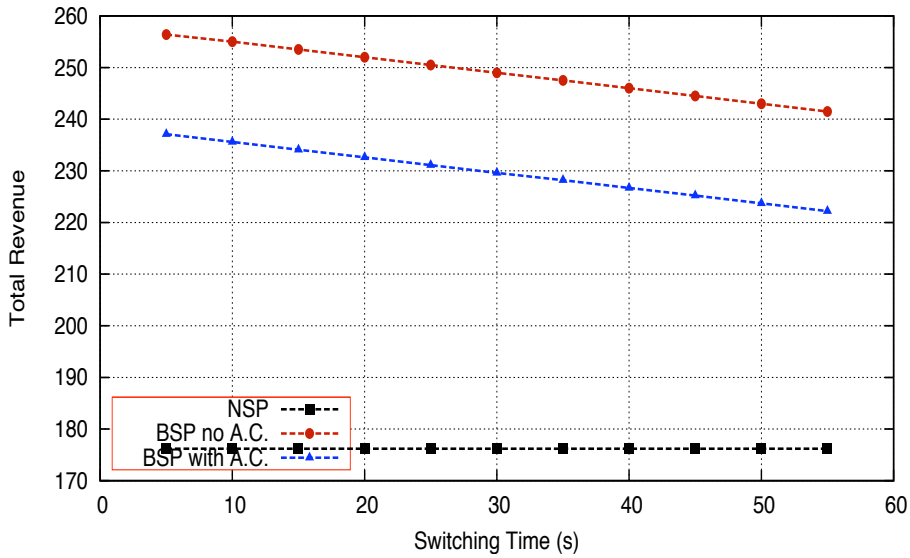


Figure 3.8: Revenue Generated by the Bottleneck Aware Switching Policy (BSP) Over Workload Two at Different Reallocation Times

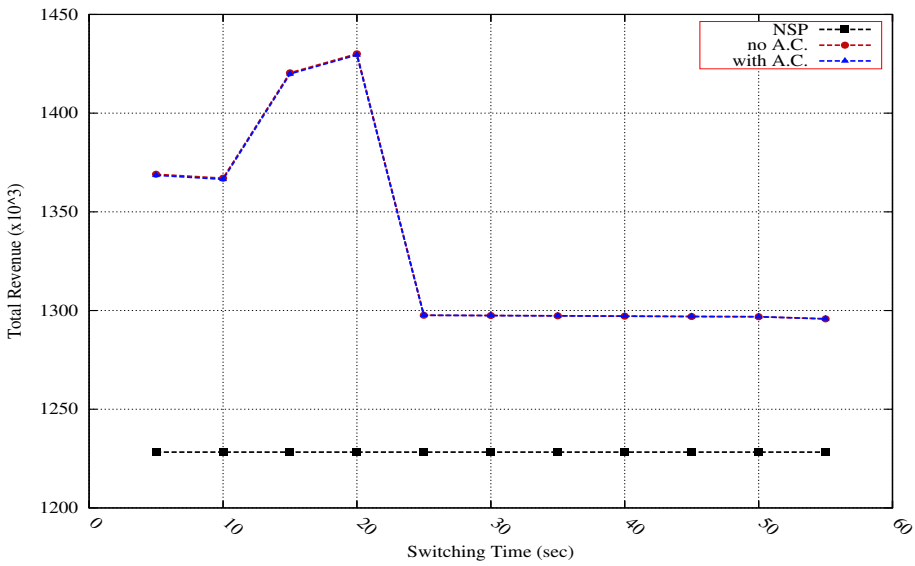


Figure 3.9: Revenue Generated by the Proportional Switching Policy (PSP) Over Workload Three at Different Reallocation Times

The number of switches servers when the PSP is applied with and without AC remains the same (just one single switches) with the increase of the switching times. The same behaviour happens when the experiments are conducted over the seconds workload with and without the admission control (AC) but the change is on the number of switches (there are three different switches).

### 3.5.3 Experiment Three

The results from applying the PSP and BSP over the third workload are shown in Figures 3.9 and 3.10 respectively. The workload that used in this experiment is is generated from two real-world Internet traces containing 76,086 requests in total and each of which contains a days worth of HTTP requests to the EPA WWW server located at Research Triangle Park, NC and the SDSC WWW server located at the San Diego Supercomputer Center in California respectively.

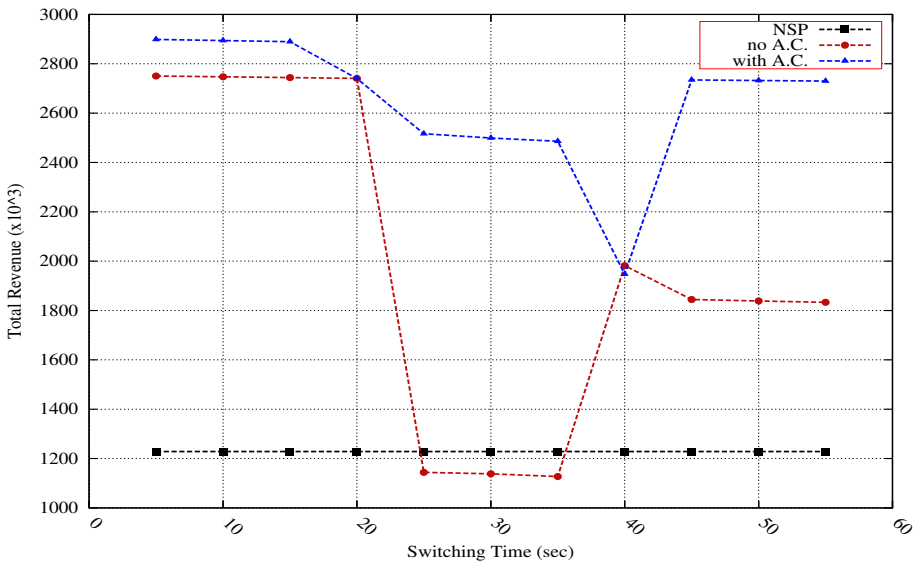


Figure 3.10: Revenue Generated by the Bottleneck Aware Switching Policy (BSP) Over Workload Three at Different Reallocation Times

Figure 3.9 shows that the system revenue can be improved by up to 16.4% when the switching time is 20 seconds when the PSP (either it is applied with or without AC) is applied over the third workload. There is a significant reduction in the system revenue when the switching time is selected to be 25 seconds, after which the revenue starts to drop until it reaches the lowest value when the switching time is 55 seconds (5.5% improvement with or without AC). The number of servers switches when PSP is applied over the third workload (either with or without AC) starts with the value of 18 switches when the switching time is 5 and then decreases until it reaches 14 when the switching time of 20 seconds is used, then drops to 13 when the switching times are 25, 30, 35, 40, 45, and 50 seconds.

Figure 3.10 shows that the system revenue is decreased by %6.9 and 7.3%, and 8.2% when the wrong switching times are selected (25, 30, 35 seconds) when BSP is applied without admission control. The number of server switches in this situation is

16, 16 and 14 respectively.

In contrast, applying the admission control always guarantees a positive improvement in the system revenue when it is applied alongside BSP over the third workload. The minimum improvement that can be obtained when applying BSP with AC over the third workload is 58.5%, when the switching time is 40 seconds (the number of switches is 16), while the maximum improvement (136%) can be obtained when the switching time is 5 seconds and there are just seven switches.

### 3.5.4 Experiment Four

Figure 3.11 and 3.12 shows the results from applying the switching policies PSP and BSP with and without the admission control (AC) over the fourth workload which has been collected from ClarkNet WWW server for the Metro Baltimore-Washington DC area. It contains 3,328,587 requests issued to the server during the period of two weeks.

The Figure 3.11 shows that the system revenue (PSP with and without the AC) improves with the increase of the period of the switching time from 5 seconds to 55 seconds. As compared with the NSP, the system revenue improves by 13.9% and 10.5% when the switching time is 5 seconds to reach up to 30.3% and 22.1% when the switching time is 55 second when the PSP is applied with AC and without AC respectively. The number of server switches when the PSP is applied along side the AC increases gradually from 155 switches when the switching time is 5 seconds, to 261 switches when the switching time is 55 seconds.

The number of server switches when the PSP is applied with no AC has the same behaviour as when the PSP is applied with AC, with just a change in the number of server switches (there are 252 switches when BSP is applied with no AC; when the

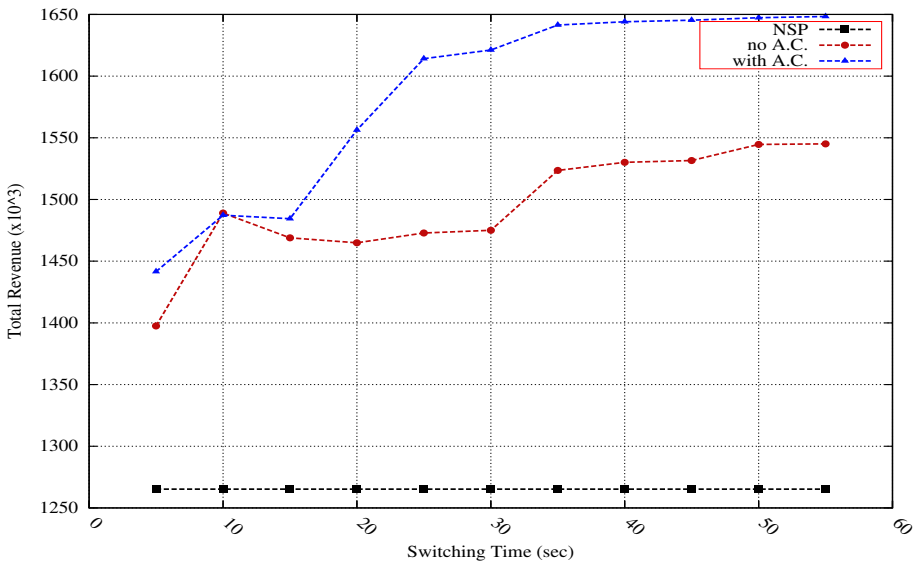


Figure 3.11: Revenue Generated by the Proportional Switching Policy (PSP) Over Workload Four at Different Reallocation Times

switching time is equal to 5 seconds, this number increase gradually to reach 385 switches where the switching time is equal to 55 seconds).

The results from applying the switching policy (BSP) over the fourth workload alongside the AC and without AC are shown in Figure 3.12. The figure shows that applying BSP over the fourth workload always improves the system revenue, either when it is applied alongside AC or with no AC by at least 246.7% and 131.5% when the switching time are 50 second and 30 seconds respectively comparing with the situation where no switching policy is applied (NSP). The maximum improvement can be obtained from applying the BSP over the fourth workload are 253.5% (where the switching time is 55 seconds) and 226.7% (where the switching time is 5 seconds) when it is applied with AC and without AC respectively.

The number of server switches where the maximum improvement is obtained are the following; 15 switches and 18 switches with AC and without AC respectively. On

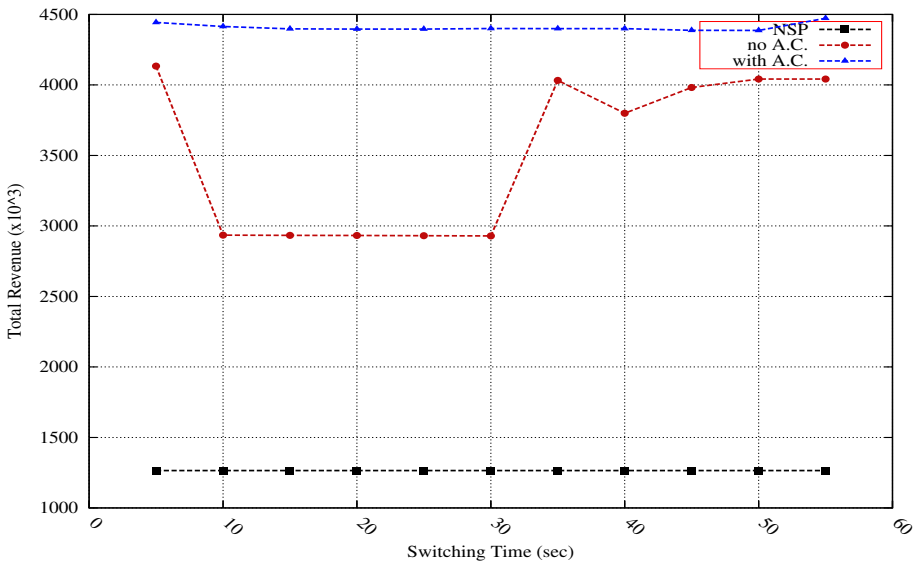


Figure 3.12: Revenue Generated by the Proportional Switching Policy (BSP) Over Workload Four at Different Reallocation Times

the other hand there are 15 switches and 44 switches when the minimum improvement in the system's revenue is obtained with AC and without AC respectively (where the switching times are 55 seconds and 5 seconds).

The number of server switches when the BSP is applied alongside the AC increase from 14 switches when the switching time is 5 seconds to be 44 switches when the switching time is 45 seconds, and after that there is reduction in the number of switches to be 15 switches when the switching time changes from 45 seconds to 50 seconds and also when it is 55 seconds.

The number of server switches when the BSP is applied with no AC has the same behaviour when the BSP is applied with AC with just two exceptions. The first concerns the number of switches, as it starts with 18 switches when the switching time is 5 second and reaches 46 switches when the switching time is 50 seconds; after that 19 server switches are enacted with a switching interval of 55 seconds. The second ex-

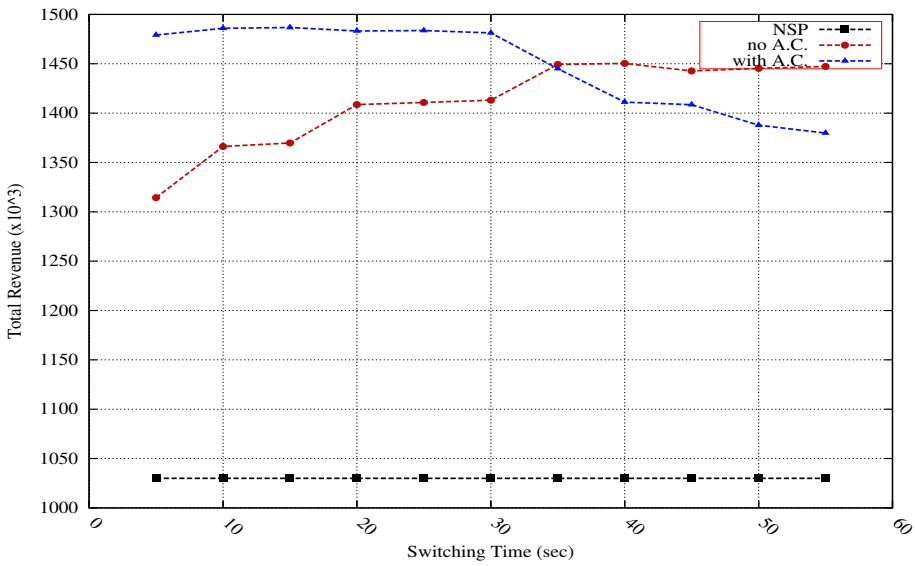


Figure 3.13: Revenue Generated by the Proportional Switching Policy (PSP) Over Workload Five at Different Reallocation Times

ception happens in the behaviour of the number of switches, as there is a sudden drop in the number of server switches when the switching time changes from 15 seconds (from 20 switches) to 20 seconds (to just 6 switches), the number of switches returns to 38 switches where 25 seconds is selected to represent the switching time.

### 3.5.5 Experiment Five

The fifth workload that the switching policies (PSP and BSP) are applied in this experiment is obtained from the NASA Kennedy Space Center web-server in Florida [2], it contains 3,461,612 requests spanning two months and the results are shown on Figures 3.13 and 3.14.

Figure 3.13 shows the results from applying the switching policy (PSP) over the fifth workload with AC and with no AC. The figure shows that the average improvements from applying PSP with AC are between 34% - 44.3% compared with NSP.

While between 27.6% - 40.8% an improvement in the system revenue can be obtained when the experiment is conducted using the PSP with no AC over this workload using the proposed switching policies. The figure also shows that the system revenue from applying the PSP with AC decrease when the switching times increased from 5 seconds to 55 seconds, while it increases when the PSP with no AC when the switching times changed from 5 seconds and 55 seconds.

The maximum improvement from applying the PSP with AC and no AC are obtained when the switching times are 15 second and 40 seconds, where the number of server switches are 246 switches and 510 switches respectively. While the number of server switches are 607 switches and 367 switches when the least improvements are obtained from applying the PSP with AC and without AC over the fifth workload. In general the number of switches when PSP is applied with AC and without the AC over the fifth workload are increased dramatically from 242 when the switching time is 5 seconds switches to be 607 when the switching time is 55 seconds and from 367 switches when the switching time is chosen to be 5 second to 917 switches when 55 is chosen to represent the switching time respectively.

The results from applying the BSP over the fifth workload with AC and without AC compared with the NSP is shown in Figure 3.14. The figure shows that applying the admission control alongside BSP does not improve the system revenue over this workload with just one exception, when the switching time is 40 seconds. The results show that the number of server switches for both scenarios are increased with the increase of the switching time; from (3 switches, 5 second switching time) to (15 switches, 55 second switching time) when BSP is applied using the admission control, and from (4 switches, 5 switching time) to (36 switches, 55 seconds switching time) when it is applied without using the admission control.

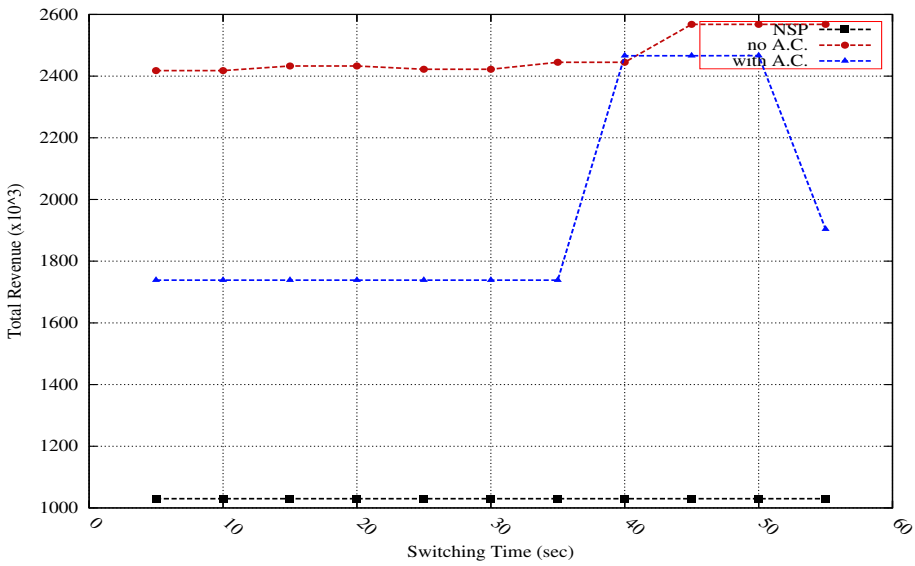


Figure 3.14: Revenue Generated by the Proportional Switching Policy (BSP) Over Workload Five at Different Reallocation Times

### 3.5.6 Experimental Results Analysis

Essentially we contrast an enterprise system with a dynamic server switching policy (PSP or BSP) and a system that uses different time for allocating the several resources between the applications pools. The experiments have been conducted over five different workloads. There are some interesting observations from the results that obtained from applying the switching policies over the first two synthetic workload;

- the reduction in revenue due to increased reallocation intervals generally holds over all server switching policies, with some exceptions;
- it has been found that minimising the reallocation interval is therefore crucial;
- therefore, optimising the reallocation interval will be application specific;
- applying the admission control (AC) policy does not always improve the system

performance especially over a light workload.

The following results can be noted from the experiments that have been conducted over the three real-world Internet traces;

- the reduction in the switching times does not always leads to an improvements in the system revenue;
- in some scenarios the system revenue is improved gradually with increasing the periods of the switching time (e.g. see Figure 3.11);
- applying the admission control does not always come with an improvement in the revenue of the system when it is applied alongside the switching policies over the real-world Internet traces (see Figures 3.13 and 3.14);
- the results that are obtained from the conducted results do not show any relationship between the number of server switches and the system revenue;

## 3.6 Summary

In this chapter we have (1) modelled an internet service provider as a collection of multi-class closed queueing networks, each of which represents a three tier web application architecture with a cluster of servers at each tier. Our model supports the dynamic reallocation of servers at the same tier between pools, (2) evaluated the behaviour of two switching policies and compared them against a pre-defined allocation over a range of reallocation intervals and observed that larger reallocation intervals have a negative impact on revenue, (3) found that the results over the used workload traces demonstrate a clear inversely proportional relationship between reallocation interval and revenue with some exceptions when the switching policies are applied over

the first two synthetic workload, while on the other hand this relationship does not always true when the experiments are applied over the real-world Internet traces, (4) the experiments show that there is no direct correlation between the number of server switches and the improvement to the system revenue.

Current switching policies evaluate the system retrospectively and make no predictions about the workload after making a migration. In the next chapter, the *reactive* behaviour of the two switching policies – Proportional Switching Policy (PSP) and Bottleneck-aware Switching Policy (BSP) – will be combined at the *proactive* properties with several forecasting algorithms in order to investigate better guide policy decisions based on real-world Internet traces from several sources.

# Chapter 4

# Predictive and Dynamic Resource Application for Enterprise Applications

## 4.1 Introduction

Dynamic resource allocation systems have been shown to improve revenue for enterprise systems by reallocating servers into a more beneficial configuration; contrast this with pre-defined systems which are periodically unable to deal with significant changes in workload, and as a result lead to a decrease in revenue.

As the use of these applications becomes more widespread, so issues concerning infrastructure performance and dependability become more significant. It is widely recognised that a slow or unreliable response from an e-Business site is one of the main reasons for a customer to seek an alternative [48].

Such issues are mitigated through capacity planning and workload forecasting [53]. However, forecasting is error prone – one need only look at recent examples from the financial markets, climate studies or production and operations management [34] to be aware of these concerns.

The observation of past values in order to anticipate future behaviour represents

the essence of the forecasting process as seen in this chapter. Numerous predictors are discussed and the way in which they are applied in the context of dynamic resource allocation is analysed. Our premise is that workload forecasting may assist revenue-generating enterprise systems which already employ methods of dynamic resource allocation; however, as with forecasting in other domains, the predictions may in fact be wrong, and this may result in server reallocation to the detriment of the service.

Two well-known methods are used for data collection in web analytics [33]. In the first, called *server-side data collection*, the log files where all transactions and requests to the web site are stored undergo systematic analysis; in the second, the visitor's Web browser is used to collect data. In this thesis we employ the first method, that is collecting the data directly from the web server.

The workload can be characterised at four different levels: the business layer, the session layer, the function layer, and the HTTP-request layer [38]. Here the real-world Internet workload is characterised at the second of these levels, where the set of requests issued from different users are clustered periodically.

Workload forecasting approaches can be divided into two different categories: quantitative and qualitative [40]. The qualitative approach is a subjective process based on different information services such as expert opinion, historical analogy, and commercial knowledge. The estimation of future values of different workload parameters which relies on the existence of historical data, i.e. that seen in the quantitative approach, is the approach to forecasting used in this thesis.

The predictive forecasting is based on past values, using several different (but common) predictors: Last Observation (LO), Simple Average (SA), Sample Moving Average (SMA), Exponential Moving Average (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average (ARIMA). In this chapter, these forecasting

algorithms are combined with our two switching policies – the proportional switching policy (PSP) and the bottleneck aware switching policy (BSP). Thus our contribution is that the forecasting and switching work in tandem; after applying the predictor, the system's resources are reallocated with respect to the prediction.

### 4.1.1 Chapter Contributions

The specific contributions of this chapter are as follows:

- We build on the model-based environment in Chapter 3 so that the combination of workload prediction and dynamic server switching can be explored. A multi-tiered, cluster-based, multi-server solution is again used; we also employ bottleneck identification through the use of convex polytopes and also admission control for dealing with the issue of overloading;

- After introducing several schemes for workload prediction in this context, including Last Observation (LO), Simple Average (SA), Sample Moving Average (SMA), Exponential Moving Average (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average (ARIMA), the forecast accuracy of these schemes is compared using Mean Square Error (MSE), Mean Average Percentage Error (MAPE), Mean Absolute Deviation (MAD) and the Cumulative sum of Forecast Error (CFE) approaches. The proportional switching policy (PSP) and the bottleneck aware switching policy (BSP) are then employed in the context of these seven workload prediction algorithms. All fourteen cases are compared with a control system where no switching policy (NSP) is applied;

- We base our results on real-world workload traces from several sources, including from the San Diego Supercomputer Centre, from the ClarkNet Internet ac-

cess provider for the Metro Baltimore-Washington DC area and, from the NASA Kennedy Space Center web-server in Florida.

## 4.2 Additional Related Work

This work is different from that found in the previous chapter in the following respects. First, the experiments are conducted using the same switching policies (Proportional Switching Policy and Bottleneck-aware Switching Policy), but with additional model-based workload prediction. Second, the workload used is also based on real-world Internet traces obtained from three different sources and containing one day, two weeks, and two months worth of HTTP requests (see Section 4.4 for more details).

The use of five different predictive algorithms (regression method, linear regression, nonlinear methods, moving average, and exponential smoothing) are all found in [40] to enable workload forecasting for Web services. Several of the predictors found here (last observation, sample average, low pass filter, and ARIMA model) were used in [23] to predict the behaviour of data-exchange in the Globus Grid middleware MDS.

To the best of our knowledge this is the first example of a model-based workload prediction and dynamic server switching analysis in the context of real-world HTTP requests.

As in previous capacity planning work [7][20][40], we generate a workload model from the characterisation of real data.

## 4.3 Modelling of Multi-tiered Internet Services and Server Switching Policies

The system model developed in Chapter 3 is reused in this chapter:

- the number of applications and servers are as found in Table 3.2;
- the PSP and BSP switching also reused in this chapter,
- the technique that used the convex polytopes approach to identify bottlenecks is applied in this chapter;
- admission control is a solution to the overloading problem, it is also used here.

## 4.4 The Workload and Predictive Algorithms

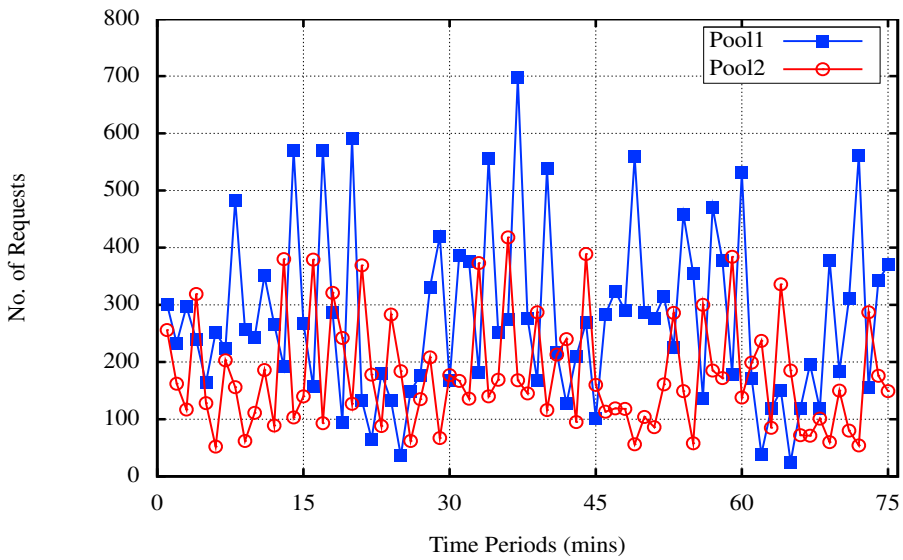


Figure 4.1: A sample of the total requests in the real-world workload for both application pools

### 4.4.1 The Workload

The workload is defined as the set of all inputs the system receives from its environment during any given period of time [40]. In this study the workloads (e.g. see Figure 4.1) are based on three real-world Internet traces containing two days, two weeks,

and two months worth of HTTP requests. The first workload is generated from two real-world Internet traces containing 76,086 requests in total and each of which contains a days worth of HTTP requests to the EPA WWW server located at Research Triangle Park, NC and the SDSC WWW server located at the San Diego Supercomputer Center in California respectively [27]. The second workload has been collected from ClarkNet WWW server which is a full Internet access provider for the Metro Baltimore-Washington DC area [2]. This workload contains 3,328,587 requests issued to the server during the period of two weeks. The third workload used in this research is obtained from the NASA Kennedy Space Center web-server in Florida [2]. This trace contains 3,461,612 requests spanning two months.

These are three real-world Internet traces are chosen based on the availability of such data and the difficulty of obtaining such new data from different sources as a results of the sensitivity of the such data. We also choose 'open data' as it allows (i) our results to be reproduced by other scientists, and (ii) it allows others to build on our own findings.

#### 4.4.1.1 Workload Characterization

Since real workload is usually unpractical, very complex and contains thousands of different programs, transactions and requests, it is unrealistic to consider the use of real workload in capacity planning studies [20], [7], and [40]. Thus the process of workload characterisation is applied to the workload in order to generate the workload model that captures the most relevant characteristics of the real workload [40]. Then the workload model can be used for different purposes e.g. performance tuning and capacity planning.

Understanding the workload is essential to performing the characterisation process

and in this situation all the three trace data, which are called log files, contain lines with the following information; host field which contains the client IP address (e.g. dcs.warwick.ac.uk), identification field to show the user identity, login name contains the authorisation user ID, timestamp contains date, time and zone, request field to show the type of the request (e.g. GET), the name of the resource (e.g. /img.gif), and the protocol version in use (e.g. HTTP/1.0), status field which specifies the response status from the server (e.g. 200), and file size field which indicated the number of bytes transferred in response to a file request (e.g. 3185). The identification and login name fields are usually hidden for security reasons.

In typical fashion (see also [40]) we characterise this workload to form a workload model, which can then be used as the input to our system model. We consider a typical Web farm as in [52], which records the hypertext transfer protocol (HTTP) service requests and aggregates them over small time intervals of length $\Delta > 0$ to obtain a time series. In this work the period of 5 minutes has been chosen to represent the interval time over one day, two weeks, and one month periods giving a total of 288 intervals in a day (see example in [52]).

### 4.4.2 Predictive Algorithms

There are often a wide variety of predictive algorithms available either based on classical time-series analysis or data-mining techniques. The different parameters used in the predictors are summarised in Table 4.1

Several predictive algorithms are employed in this work based on classical time-series analysis:

Table 4.1: Notation used in the predictors

| Symbol | Description |
|---|---|
| $P_x$ | The predictive value |
| $P_{x-1}$ | The previous predictive value |
| $V$ | The last actual value |
| $s$ | The sample set size |
| $\alpha$ | The weighting factor |

#### 4.4.2.1 i) Last Observation (LO)

The forecasting procedure is based on the most recent observation. The last value ($V_{x-1}$) is most likely to reflect the behaviour of future queries ($P_x$) [23].

$$P_x = V_{x-1}$$

#### 4.4.2.2 ii) Simple Algorithm (SA)

The simple average algorithm is appropriate for short term forecasting [40]. The accuracy achieved by the technique is usually high when it is applied to nearly stationary data [28]. In this algorithm the predictive value $P_x$ is the mean average of all the previous observations.

$$P_x = \frac{\sum_{i=0}^{x-1} V_i}{x-1}$$

#### 4.4.2.3 iii) Sample Moving Algorithm (SMA)

The predictive value $P_x$ is the mean average of the past performance values within a sample set $s$. Equal weighting is given to each performance value in this predictive algorithm.

For this study we set the sample set $s$ to be of size 3; other set sizes can be easily tested within this proposed framework.

$$P_x = \frac{\sum_{i=x-s}^{x} V_x}{s}$$

#### 4.4.2.4 iv) Exponential Moving Algorithm (EMA)

In this predictor, an older value within the sample set is given less importance than a newer value, This is done by applying a weighting factor (which declines exponentially) for each value in the set.

$$P_x = P_{x-1} + \alpha \times (V - P_{x-1})$$

#### 4.4.2.5 v) Low Pass Filter (LPF)

The low pass filter also weights recent data more heavily than older data, where the weight on each observation decreases exponentially by the number of observations using the following formula:

$$P_x = (w \times P_{x-1}) + ((1 - w) \times V)$$

Here $w$ represents the weighting parameter and its value's between 0 and 1. If the value of $w$ is equal to 0, then the low pass filter is the same as the last observation (LO). On the other hand, the filter never changes if $w = 1$. In terms of increasing the accuracy of the low pass filter prediction, the value of the weighting parameter $w$ is set to 0.95, see [14] for more details.

#### 4.4.2.6 vi) Autoregressive Integrated Moving Average Model (ARIMA)

The Autoregressive integrated moving average model ($ARIMA$) is, in theory, the most general class of model for forecasting a time series which can be stationarized by transformations such as differencing and logging [42], where the differenced series appearing in the forecasting equation are called ”auto-regressive” terms, the forecast errors are called ”moving average” terms, and a time series which needs to be differenced to be made stationary is said to be an ”integrated” version of a stationary series.

Here in the autoregressive integrated moving average model ($ARIMA(p,d,q)$); $p$ indicates the order of the autoregression, $d$ indicates the amount of differencing, and $q$ indicates the order of the moving average part. The experiments have been conducted using two well-known ARIMA models. The first model is defined as AR(1) where the forecasting process for the next value is based on the value in the last time period; when based on the last two values in the previous two time periods, this is termed AR(2).

The work in [11] shows in details of how to predict the next-day electricity prices in the mainland Spanish and Californian markets, respectively based on the ARIMA methodology. However, there are various available packages that apply the methodology (Box-Jenkins), and parameter optimisation, in order to find out the most accurate parameters for the ARIMA model (e.g. *R* project and IBM SPSS Statistics).

## 4.5 Experimental Setup and Results

### 4.5.1 Experimental Setup

As in Chapter 3, two applications are modelled as running on two logical pools. Each of these is multi-tiered, with each tier comprising a cluster of servers. The system developed in this work contains just two application pools and the servers are reallocated

between these two pools. This system model is used for the sake of simplicity and in particular to better understand the general characteristics of our findings; clearly more application pools can be used, and this work is applicable to systems of greater size. We believe that the results from this study are more generally applicable, however, this remains the topic of further research.

The service time $S_{ir}$, the visiting ratio $v_{ir}$ and the remaining experimental parameters are based on realistic (i.e. sampled) values, or from those supplied in supporting literature [54].

Different measures to assess forecasting accuracy have been applied; this is done by calculating the predicted values from several different predictive algorithms and comparing these with the real-world data. Various accuracy measures have been used in the literature and their properties are well understood. The forecast accuracy measures that have been used here are: Mean Square Error (MSE), Mean Average Percentage Error (MAPE), Mean Absolute Deviation (MAD) and the Cumulative sum of Forecast Error (CFE).

Table 4.2 describes the supporting mathematics for each of these different forecast accuracy measures. Additional notation used in these equations – $N, O, and P$ – represent the data sample set size, an observed value and a predicted value respectively.

### 4.5.2 Accuracy Forecasting Results

The resulting values from each of the forecast accuracy measures (MSE, MAPE, MAD, and CFE) over the three real-world Internet traces, with respect to the observed performance values, are shown in Tables 4.3, 4.4, and 4.5. In each case we are looking for a value as close to zero as possible.

Table 4.2: Forecasting accuracy measures

| Forecast accuracy measures | Equation |
|---|---|
| Mean Squared Error (MSE) | $\frac{1}{N}\sum_{i=0}^{N-1}(O_i - P_i)^2$ |
| Mean Absolute Percent Error (MAPE) | $\frac{1}{N}\sum_{i=0}^{N-1}\left[\frac{\lvert O_i - P_i\rvert}{O_i}\right] \times 100$ |
| Mean Absolute Deviation (MAD) | $\frac{1}{N}\sum_{i=0}^{N-1}(\lvert O_i - P_i\rvert)$ |
| Cumulative Sum of Forecast Errors (CFE) | $\sum_{i=0}^{N-1}(O_i - P_i)$ |

Thus, with regard to the MSE over the first workload, we see that LPF has the highest value and AR(1) has the lowest value, which means that the most accurate predictor based on the MSE is AR(1) and the least accurate predictor is LPF (see Table 4.3).

SMA, EMA, and AR(2) perform well when MAPE and MAD are applied to determine the forecast accuracy, AR(1) however wins out in this case. The least accurate predictors according to these two policies (MAPE and MAD) over the first workload is LPF. Again LPF is also the least accurate and AR(1) the most accurate with CFE over the first workload. It can be said that, Table 4.3 shows that the AR(1) has the lowest values with all four predictors and LPF has the highest values. In other words, respectively the AR(1) and LPF are the most accurate and least accurate predictors over the first workload according to the four different accuracy forecasting policies.

LPF is the most accurate predictor with MSE over the second workload and the SA is the least accurate one (see Table 4.4). LPF is the best predictor when MAPE and MAD are used to testify the forecast accuracy of the different predictors. The least accurate predictor is SA with MAPE and MAD; AR(2) and AR(1) are the most and least accurate predictors respectively with CFE.

With regard to the MSE, it can be seen from Table 4.5, that SA has the highest value and AR(1) has the lowest value, which means that the most accurate predictor is AR(1) and the least accurate predictor is SA. LPF and AR(1) perform well when MAPE and MAD are applied to determine the forecast accuracy respectively, while the least accurate predictors are SA with these two accuracy measures (MAPE and MAD). EMA is the most accurate and LPF the least accurate predictors with CFE over the third workload.

### 4.5.3 Accuracy of the Forecasting Analysis

Table 4.6 summarises the results obtained from applying the four different accuracy forecasting metrics – MSE, MAPE, MAD, and CFE – over the three real-world workloads used in this work. The table shows the most and least accurate predictors in each case. The table shows that according to the first accuracy measures used (MSE), the AR(1), LPF, AR(1) are the best predictors when they are applied over the first, second, and third workload that are used respectively, while LPF, AR(1), and SA are the worst predictors over these three workloads. The mean average percentage error (MAPE) shows that the LPF algorithm is the best predictor when it is applied over the second and third workloads, but it does not work well when it is applied over the first workload in this experiment (AR(1) is the best algorithm over the first workload). Over the second and third workload the predictor SA is the worst algorithm according to MAPE and MAD measures.

According to the third accuracy measure that is used in this experiment, the mean absolute deviation (MAD), AR(1) is the most accurate predictor when it is applied over the first and third workloads, while the LPF is the best predictor over the second workload. LPF however is the least accurate prediction algorithm over the first

workload, and SA is the least predictor over the remaining workload that is used in this research. While the results based on the last accuracy measure that is used in this research (CFE) over the three workloads show that; AR(1), AR(2), EMA are highly recommended to be used over the three workloads respectively, while LPF is the least accurate prediction algorithm when it is applied over the first and third workload. Although AR(1) is highly recommended to be used overt the first workload according to CFE, it is however the worst predictor when it is applied over the second workload.

In conclusion it can be seen from the table that:

- However the results for the first workload are consistent based on using the four accuracy measures (MSE, MAPE, MAD, and CFE) show that AR(1) is the most accurate predictor and LPF is the least accurate according to our metrics.
- This said, this same pattern – (AR(1) is the best and LPF is the least accurate predictors – does not hold for the second and third workload, where the results are more mixed as explained early. Notably, LPF performs best in some cases.
- We conclude that no one scheme is preferable over any other, and this motivates our work in the reminder of the chapter. In other words, the results show that the accuracy measure can not be used as a tool to choose the most accurate predictor over the used workloads and new techniques need to be developed to over come the inconsistent of the results that obtained from the different accuracy measures that used in this research (MSE, MAPE, MAD, and CFE).

Next we combine forecasting with dynamic server switching, and explore the likely benefits of this combination.

### 4.5.4 Combining Forecasting and Dynamic Server Switching

The system gain in revenue from applying several different predictive algorithms with the two server switching server policies over the three workloads are shown in Tables 4.7, 4.8, and 4.9. In each case the results show the base-line revenue when no switching policy is applied (NSP) and also the case when the switching policy alone (without forecasting) is applied. These provide good indicators against which the new results can be compared.

### 4.5.5 Experiment One

The first experiment is conducted using the first workload which has been generated from two real-world Internet traces to the EPA WWW server located at Research Triangle Park, NC and the SDSC WWW server located at the San Diego Supercomputer Center, California respectively [27]. The results from the different predictive models based on this workload are shown in Table 4.7. This table represents the gained revenue from applying the seven predictive algorithms – LO, SA, SMA, EMA, LPF, AR1, and AR2.

The two server switching policies used in this work (PSP and BSP) provide 5.63% and 103.47% improvement in system revenue compared to the non switching policy (NSP). The results from Table 4.7 show a 12.15% improvement in the system revenue when the AR1 algorithm is applied with PSP, compared to that from the original PSP without prediction. The remainder of the predictive algorithms also yielded better results than the original revenue computed from the PSP scheme without forecasting, with just two exceptions – when the predictors SA and SMA are used the revenue drops by -0.12% and -0.46% respectively.

Table 4.7 also show that applying the predictive algorithms with BSP provides

better results (revenue) than the original BSP without prediction. It provides at least 6.67% improvement in system revenue with LPF and up to 8.89% when EMA is applied alongside BSP.

### 4.5.6 Experiment Two

We repeat the experiments with the second workload – this contains 3,328,587 requests issued to the ClarkNet WWW server over the period of two weeks. The resulting values are shown in Table 4.8. As can be seen from Table 4.8, the different predictors that have been applied to the proportional switching policy (PSP) do not always provide better results (revenue) than the original PSP without prediction. The improvement in system revenue dropped by 0.83%, 3.02%, and 6.41% when the predictors SMA, LO, and SA are applied along with PSP respectively. However, the system performance can be improved by at least 0.05% when the EMA is applied with PSP and up to 1.78% when AR(1) is applied with the PSP. The rest of the predictors (LPF and AR(2)) bring an improvement to the system by 0.68% and 1.27% respectively.

Table 4.8 also shows that the revenue improvement from applying the different predictors with BSP does not always exist as it is dropped by 0.79% with LPF and 2.03% with AR(1) comparing with that from applying the BSP without prediction. The system's performance can be improved by 7.28% when the SMA is used with the BSP. The range of positive revenue improvement is between 3.33% and 4.96% when the rest of the predictors are applied with BSP.

### 4.5.7 Experiment Three

Finally we undertake the same experiments using the third workload, which contains 3,461,612 requests to the the NASA Kennedy Space Center web-server in Florida.

Results are shown in Table 4.9. It is shown that the revenue of the system can be improved by as much as 69.19% and 44.14% when BSP and PSP are applied to the system compared with the NSP; a further 45.91% and 0.76% improvement can be achieved when SA and AR(1) are applied to the system along with BSP and PSP respectively (see Table 4.9).

Table 4.9 shows that, the different predictors that have been applied alongside the proportional switching policy (PSP) do not always provide better results (revenue) than the original PSP without prediction, as when the predictors AR(2), LPF, LO, and SA are used, the revenue drops by -0.01%, -1.34%, -3.05%, and -5.06% respectively. Nevertheless all the predictors including AR(1) perform better than the non switching policy (NSP).

It can be seen also that from Table 4.9 the revenue improvement from applying different predictors with BSP is at least 8.13% over that when the BSP is applied to the system without prediction. The range of revenue improvement is between 8.13% and 38.77% when the LO (8.13%), EMA (12.54%), AR(1) (24.64%), SMA (33.97%), and AR(2) (38.77%) are applied with BSP to the system. In addition to this, the revenue is improved by 45.02% when LPF is applied with BSP, and further 0.89% improvement when SA is applied to the system along with BSP.

### 4.5.8 Experiments Results Analysis

Table 4.10 shows the results obtained from applying the seven different predictors along with the two switching policies (PSP and BSP) over the three selected workloads, compared with that obtained from the switching policies with no prediction. It can be seen from the table that:

- AR(1) is the most accurate predictor over the first, second, and third work-

load, it can provide up to 12.15%, 1.78%, and 0.76% respectively when it is applied along with PSP compared with applying PSP with no prediction. However AR(1) is the least accurate predictor when it is applied with BSP over the second workload (-2.03%);

- It can be seen that, SA is the best predictor when it is applied with the BSP over the third workload (SA, the third workload, BSP, 45.91%). It brings however the least performance when it is applied with PSP over the second workload (SA, the second workload, PSP, -6.41%);
- The system revenue can be improved by up to 45.91% (BSP, SA, NASA workload) compared with that obtained from applying the BSP with no prediction. These are significant revenue gains;
- The average improvement that can be obtained from applying the best selected predictors along with PSP and BSP over the three workload are 5% and 21% respectively.

Table 4.3: Forecast accuracy, against four different criteria, for the seven forecast algorithms over the first workload

| Policy | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
|---|---|---|---|---|---|---|---|
| MSE | 5.73 | 51.61 | 0.13 | 0.27 | 81.72 | 0 | 0.38 |
| MAPE | 0.09 | 0.28 | 0.01 | 0.02 | 0.36 | 0 | 0.02 |
| MAD | 0.14 | 0.43 | 0.02 | 0.03 | 0.54 | 0 | 0.04 |
| CFE | -40 | 120 | 6 | -8.62 | -151 | 1 | -10.29 |

Table 4.4: Forecast accuracy, against four different criteria, for the seven forecast algorithms over the second workload

| Policy | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
|---|---|---|---|---|---|---|---|
| MSE | 63631.42 | 342417.84 | 50043.17 | 48962.58 | 26948.79 | 65058.85 | 52957.80 |
| MAPE | 11.35 | 28.38 | 10.10 | 9.96 | 7.40 | 11.52 | 10.30 |
| MAD | 195.35 | 464.47 | 174.04 | 171.43 | 126.66 | 198.04 | 177.50 |
| CFE | 51 | -3568 | -1075 | 49.34 | -3246 | -33637 | 37.59 |

Table 4.5: Forecast accuracy, against four different criteria, for the seven forecast algorithms over the third workload

| Policy | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
|---|---|---|---|---|---|---|---|
| MSE | 7549.95 | 28977.21 | 6645.13 | 7068.30 | 7676.35 | 6428.77 | 10138.46 |
| MAPE | 15.67 | 32.88 | 14.70 | 15.40 | 13.64 | 14.68 | 17.17 |
| MAD | 64.97 | 129.87 | 60.06 | 63 | 60.40 | 59.72 | 72.76 |
| CFE | -194 | -126628 | -5634 | 80.40 | -179288 | 174.88 | -155359 |

Table 4.6: Analysis of forecast accuracy for the seven forecast algorithms over the three workloads

| Policy | | The first workload | The second workload | The third workload |
|---|---|---|---|---|
| MSE | Most accurate predictor | 0 (AR(1)) | 26948.79 (LPF) | 6428.8 (AR(1)) |
| | Least accurate predictor | 81.72 (LPF) | 65058.85 (AR(1)) | 28977.21 (SA) |
| MAPE | Most accurate predictor | 0 (AR(1)) | 7.40 (LPF) | 13.64 (LPF) |
| | Least accurate predictor | 0.36 (LPF) | 28.38 (SA) | 32.88 (SA) |
| MAD | Most accurate predictor | 0 (AR(1)) | 126.66 (LPF) | 59.72 (AR(1)) |
| | Least accurate predictor | 0.54 (LPF) | 464.47 (SA) | 129.87 (SA) |
| CFE | Most accurate predictor | 1 (AR(1)) | 37.59 (AR(2)) | 80.40 (EMA) |
| | Least accurate predictor | -151 (LPF) | -33637 (AR(1)) | -179288 (LPF) |

Table 4.7: Revenue gains for switching policy and forecasting combinations over the first workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 614.1 | 648.7 | 650 | 647.9 | 645.7 | 677.8 | 651.3 | 727.5 | 660.6 |
| Imp. over PSP (%) | - | 0 | 0.2 | -0.12 | -0.46 | 4.49 | 0.4 | 12.15 | 1.83 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 614.1 | 1249.5 | 1341.3 | 1333.7 | 1333.9 | 1360.6 | 1332.8 | 1346.5 | 1333.4 |
| Imp. over BSP (%) | - | 0 | 7.35 | 6.74 | 6.75 | 8.89 | 6.67 | 7.76 | 6.71 |

Table 4.8: Revenue gains for switching policy and forecasting combinations over the second workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 632.6 | 810.6 | 786.1 | 758.6 | 803.9 | 811 | 816.1 | 825 | 820.9 |
| Imp. over PSP (%) | - | 0 | -3.02 | -6.41 | -0.83 | 0.05 | 0.68 | 1.78 | 1.27 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 632.6 | 2200.1 | 2273.3 | 2306.2 | 2360.3 | 2309.2 | 2182.8 | 2155.5 | 2279.3 |
| Imp. over BSP (%) | - | 0 | 3.33 | 4.82 | 7.28 | 4.96 | -0.79 | -2.03 | 3.6 |

Table 4.9: Revenue gains for switching policy and forecasting combinations over the third workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 513.8 | 740.6 | 718.0 | 703.1 | 742.2 | 745.0 | 730.7 | 746.2 | 740.5 |
| Imp. over PSP (%) | - | 0 | -3.05 | -5.06 | 0.22 | 0.59 | -1.34 | 0.76 | -0.01 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 513.8 | 869.3 | 940.0 | 1268.4 | 1164.6 | 978.3 | 1260.7 | 1083.5 | 1206.3 |
| Imp. over BSP (%) | - | 0 | 8.13 | 45.91 | 33.97 | 12.54 | 45.02 | 24.64 | 38.77 |

Table 4.10: Analysis of the forecast algorithms over the three workloads using the two switching policies (PSP and BSP)

| | PSP | | BSP | |
|---|---|---|---|---|
| Policy | Best predictor | Worst predictor | Best predictor | Worst predictor |
| The first workload | 12.15% (AR(1)) | -0.46% (SMA) | 8.89% (EMA) | 6.67% (LPF) |
| The second workload | 1.78% (AR(1)) | -6.41% (SA) | 7.28% (SMA) | -2.03% (AR(1)) |
| The third workload | 0.76% (AR(1)) | -5.06% (SA) | 45.91% (SA) | 8.13% (LO) |

## 4.6 Summary

In this chapter we have used a model-based environment in which the combination of workload prediction and dynamic server switching can be explored. A multi-tiered, cluster-based, multi-server solution is modelled, which contains bottleneck identification through the use of convex polytopes and also admission control. A workload model is also constructed from the characterisation of real data. We investigate the behaviour of server switching policies in the context of workload predictors. Several schemes for workload prediction are explored and the forecast accuracy of these schemes is compared. An evaluation of two well known switching policies – the proportional switching policy (PSP) and the bottleneck aware switching policy (BSP) – is conducted in the context of seven workload prediction algorithms. All fourteen cases are compared with a control system where no switching policy is applied.

It has been found that the AR(1) predictor is the most accurate with PSP over the three workloads, and EMA, SMA, and SA are the most accurate predictors with BSP over the first, second, and third workloads respectively.

Interestingly the accuracy of each predictor is noticeably different from one policy to another and there is no general case where improvements in revenue can be guaranteed.

We have demonstrated that revenue can be improved by as much as 46% if the right combination of dynamic server-switching and workload forecasting are used.

The next chapter will focus on identifying a way of automatically selecting the most effective dynamic server-switching and workload forecasting strategies; this will likely depend on the configuration of the system and the nature of the workload being applied.

# Chapter 5

# The Development and Application of Meta-forecasting

## 5.1 Introduction

In Chapter 4 we introduced the notion of forecasting applied alongside dynamic server switching. This extended the concept of *reactive* dynamic server switching where changes in demand on a system would trigger the reallocation of servers to applications with several forecasting schemes that would allow the server switching to become *proactive* that is a server switch would be initiated before the demand on the system changed.

While workload forecasting was shown to improve system revenue by up to 46% in some cases, we demonstrated through the use of several different forecasting methods, that revenue improvement was not consistent. Worse than this, in some cases the addition of forecasting schemes would decrease system revenue.

In this chapter we explore the notion of meta-forecasting, where several different forecasting schemes are employed simultaneously and a collective decision is made as a result of combining individual recommendations.

### 5.1.1 Chapter Contributions

The specific contributions of this chapter are as follows:

- The three real-world workload traces continue to be used together with our multi-class closed queueing network model;
- The two dynamic server switching policies continue to be used (BSP and PSP), together with the seven forecasting schemes (including Last Observation, Simple Algorithm, Sample Moving Average, Exponential Moving Algorithm, Low Pass Filter and Autoregressive Moving Average);
- As each of the forecasting schemes has its own bias, we develop three new meta-forecasting algorithms (the Active Window Model, the Voting Model and the Selective Model) to ensure consistent and improved results;
- We show through experimentation that the meta-forecasting algorithms always improve revenue compared with just employing the dynamic server switching policies alone. In the best case improvements are in the order of 40%, in the worst case the improvements are negligible. Importantly, using the meta-forecast algorithms never degrade results.

## 5.2 Additional Related Work

The research presented here (i) employs seven model-based workload prediction algorithms and extends the infrastructure to process and respond to this data, (ii) introduces three meta-forecasting algorithms, based on the observation that one predictor alone is potentially sub-optimal, (iii) the real-world Internet traces are considerably larger than those previously explored and are taken from multiple sources.

The work in [18] applies several predictive techniques to adaptive, web-cluster switching algorithms. There are two main differences between their work and ours. First, the system model is itself quite different; we model the system as two multi-tiered applications running on two pools, where servers are moved from another (quieter) pool to deal with overloading. [18] on the other hand use a model that consists of a set of servers with a switch that allocates the incoming request to one of the servers in the web cluster within a 2-tiered architecture. Secondly, the system monitoring processes are also different; in our research we use fixed-time intervals (see Section 5.3.1); [18] monitor their system using non-fixed intervals (Adaptive Time Slot Scheduling) based on the system's request arrival rate.

Server workload is usually measured in terms of the incoming request's arrival rate (e.g. total number of operations or size of the files requested per unit time). In this research the interval for collecting the data from the Internet workload has been chosen to be five minutes, giving a total of 288 intervals in a day. This is typical in the characterisation of data from websites ranging from retail industries to finance (see [52]).

## 5.3 The Workload and Predictive Models

In this study the workloads are based on Internet traces containing two days, two weeks, and two months worth of HTTP requests. The three workloads which the experiments are conducted over are presented in the previous chapter (see Chapter 4 for more details).

The predictive algorithms – Last Observation (LO), Simple Algorithm (SA), Sample Moving Average (SMA), Exponential Moving Algorithm (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average Model (ARIMA) – are docu-

mented elsewhere (see Section 4.4.2 in Chapter 4). The new meta-forecasting models that we propose, Active Window Model (AWM), Voting Model (VM), and Selective Model (SM) are described below.

### 5.3.1 Active Window Model (AWM)

Figure 5.1 demonstrates the change in revenue that results from applying each of the seven forecasting predictors to the NASA workload under the PSP switching policy. It is clear that while there are similar trends over time, some predictors produce better results than others. It is also the case that the results are not consistent, that is, one predictor does not consistently perform better than all the others.

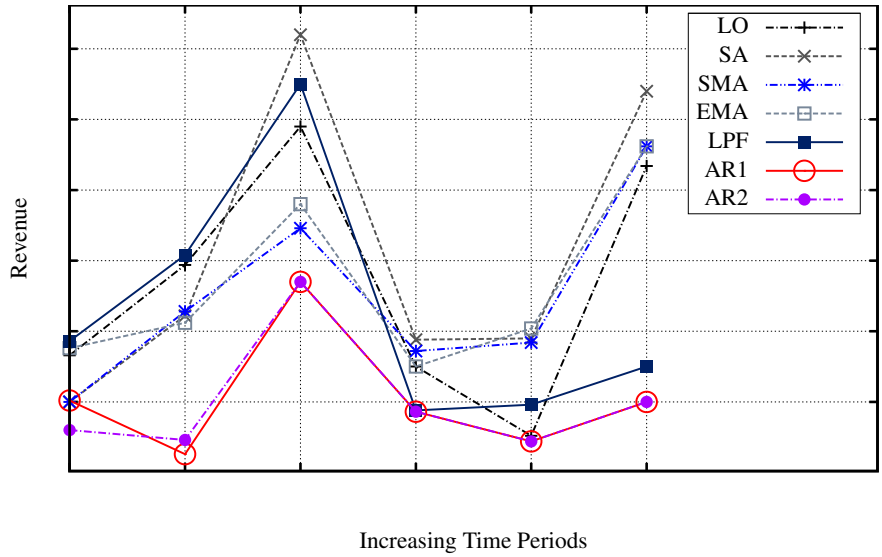


Figure 5.1: Revenue samples from applying the seven predictors (NASA workload, PSP switching policy)

In the AWM model, the data points for all predictors are collected during a fixed period (the *Active Window*). The gained revenue from each predictor is compared with the original revenue with no forecasting. The best predictor, i.e. that which results in

the highest revenue for the last period (along with the switching policy), is then used for the next period. In the case where more than one predictor deliver the same results, one is chosen at random.

In this model the active windows have varying duration: 5m, 10m, 15m, 20m, 25m, 30m, 1h, 2h, and 12h; where *m* and *h* represent minutes and hours respectively.

### 5.3.2 Voting Model (VM)

The voting model (VM) is based on the following scenario (for both switching policies). First, each of the different predictors are applied to the system, these predictions are acted upon and the system is reconfigured accordingly (resulting in one system configuration for each of the predictors/server-switching policies); the system (re-) configuration chosen most often (i.e. with the most votes) is then applied to the system proper. A random process is used where more than one configuration exists or where all predictors give different results.

This system clearly requires more calculation within the model, as we are deciding on the final state of the system as opposed to simply an up-front prediction of workload.

### 5.3.3 Selective Model (SM)

The selective model works by choosing those predictors that have performed best during the past time period, and employing these for the next time period. SM(B2) calculates the mean of the best two predictors (as compared to the original system without predictors). Several other selective models are also applied, including the selective model with the best three or four predictors (SM(B3)) and (SM(B4)), and the selective model with an average of all the predictors computed and then applied (SM(AVG)).

The random selection process is also applied in this developed prediction model

(e.g. when the SM(B2) is performed and there are three predictors performing equally, here just two of these three algorithms are selected randomly).

In each case, the choice of workload prediction technique is dynamic; that is, no one prediction technique is applied throughout the system lifetime. This aim of such an approach is to avoid bias and to ensure that the variability in the workload (which we inevitably see between the variety of input sources) is somehow accounted for. Figure 5.1 highlights the need for such a scheme; the Low Pass Filter predictor, for example, can produce the second-best revenue in one time period, to be followed by the second-worst revenue in the subsequent time period. Workload clearly impacts on the effectiveness of the prediction and dynamic server reallocation combined.

## 5.4 Experimental Setup, Results, and Analysis

The setup of the developed model along with the different parameters used in this chapter are presented in this section. The experiment results and its analysis are also discussed here.

### 5.4.1 Experimental Setup

We have developed a supporting simulator to allow us to verify the behaviour of our theoretical models. We prime the simulator with measured values from an in-house test platform, or use values from supporting literature where these are not attainable. We simulate two multi-tiered applications running on two logical pools (1 and 2) on a cluster of servers. There are two different classes of job (gold and silver), which represent the importance of these jobs. The service time $S_{ir}$ and the visiting ratio $v_{ir}$ are both based on realistic (i.e. sampled) values.

In this work, three different models have been developed to compute the request

servicing capability: 1) the Active Window Model (AWM); 2) the Voting Model (VM) and; 3) the Selective Model (SM). In each case, the results show the base-line revenue when no switching policy is applied (NSP) and also the case when the switching policy alone (without forecasting) is applied (NP). These provide good indicators against which the new results can be compared.

### 5.4.2 Experimental Results

The results from applying our new predictive models along with the dynamic server switching policies on three real-world Internet traces are shown in Tables 5.1, 5.2, and 5.3 and Figures 5.2, 5.3, and 5.4, and are described in the following sections.

#### 5.4.2.1 i) Experiment One

The first experiment is conducted using the first workload which has been generated from two real-world Internet traces to the EPA WWW server located at Research Triangle Park, NC and the SDSC WWW server located at the San Diego Supercomputer Center, California respectively [27]. The results from the different predictive models based on this workload are shown in Table 5.1 (note that this duplicates results shown in Chapter 4, but nevertheless makes a study of the new results easier). This table represents the gained revenue from applying the seven predictive algorithms – LO, SA, SMA, EMA, LPF, AR1, and AR2. In this case one or other of the predictors are applied consistently throughout the experiment. Figure 5.2 represents the revenue that has been achieved using the three meta-forecasting models – AWM, VM, and SM; this therefore represents a combination of applied predictors, depending on the details of the scheme.

The two server switching policies used in this work (PSP and BSP) provide 5.63%

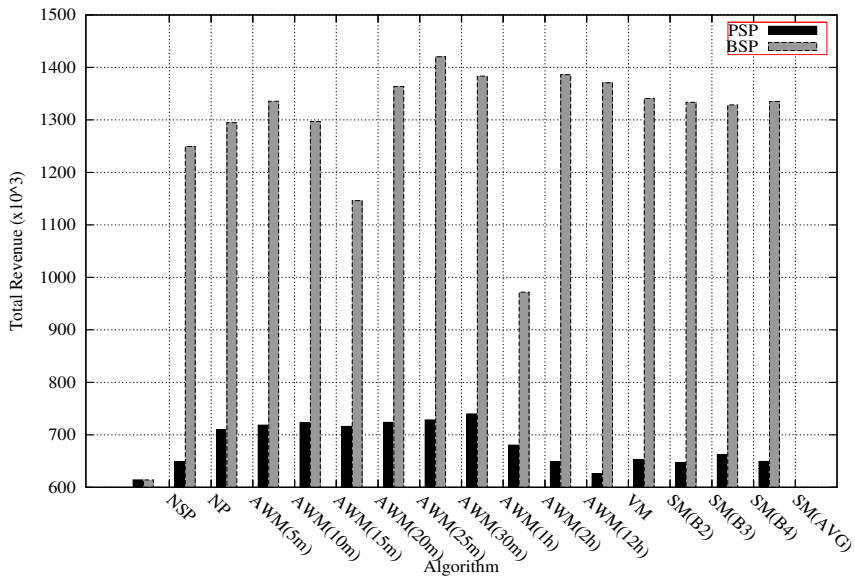


Figure 5.2: Revenue using Active Window Model (AWM), Voting Model (VM), and Selective Model (SM) over the first workload

and 103.47% improvement in system revenue compared to the non switching policy (NSP). The results from Table 5.1 show a 12.15% improvement in the system revenue when the AR1 algorithm is applied with PSP, compared to that from the original PSP without prediction. The remainder of the predictive algorithms also yielded better results than the original revenue computed from the PSP scheme without forecasting, with just two exceptions – when the predictors SA and SMA are used the revenue drops by -0.12% and -0.46% respectively.

Table 5.1 also shows that applying the predictive algorithms with BSP provides at least 6.67% improvement in system revenue with LPF and up to 8.89% when EMA is applied alongside BSP (this was displayed in Chapter 4).

The achieved revenue from using the three different meta-forecasting models is shown in Figure 5.2. The Active Window Model (AWM) performs the best on the

given workload and the revenue can be up to 14.06% higher when it is applied with PSP and up to 36.70% higher with BSP when it is applied every one and twelve hours respectively. While these results look encouraging, we sound a note of caution by highlighting the fact that the improvement drops by -0.43%, -8.27%, and -22.22% when the AWM is applied with BSP for every 10m, 20m, and 2h respectively. This method is clearly sensitive to workload and its employment and configuration therefore would need to be subject to realistic trials if it is to be most effective.

The voting model (VM) is not effective with PSP (resulting in a 3.45% drop in revenue), yet it does signify improvements with BSP (resulting in a 9.71% improvement in revenue) when its performance is compared to the switching policy with no forecasting. The Selective Model is able to bring about improvements to both switching policies, revenue is up 2.07% using a combination of SM(B4) and PSP, and is up 7.33% using a combination of SM(B2) and BSP.

#### 5.4.2.2 ii) Experiment Two

We repeat the experiments with the second workload – this contains 3,328,587 requests issued to the ClarkNet WWW server over the period of two weeks. The resulting values are shown in Table 5.2 and Figure 5.3. Table 5.2 shows that the system revenue is improved by 28.14% and 247.79% using PSP and BSP respectively compared to NSP. The figure also shows a further 1.78% and 7.28% improvement when the AR1 and SMA predictors are applied with PSP and BSP respectively (see Chapter 4).

Figure 5.3 shows that, AWM provide further improvement (by 12.24% and 8.32%) when it is applied every twelve hours with PSP and BSP on the given workload. VM decreases the performance of the system by -0.58% when it is used with PSP and up to -25.29% with BSP.

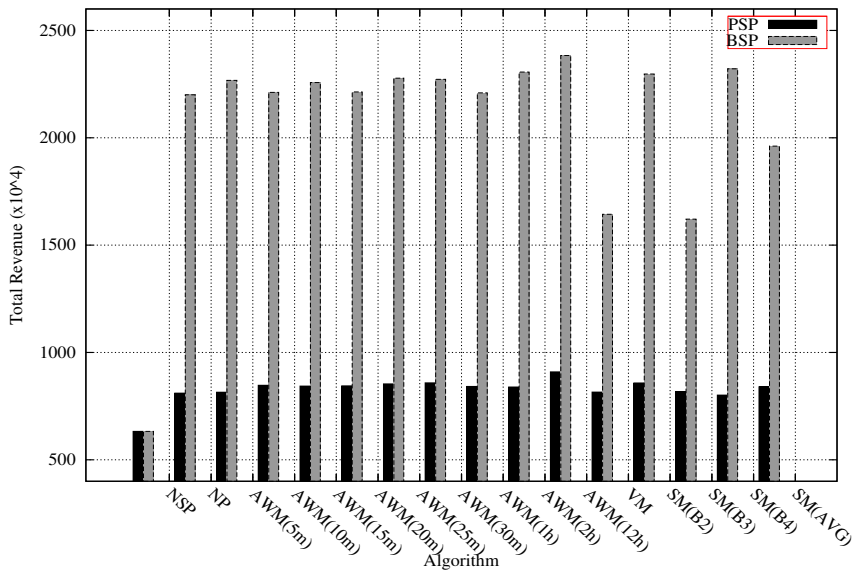


Figure 5.3: Revenue using Active Window Model (AWM), Voting Model (VM), and Selective Model (SM) over the second workload

The highest improvements that can be achieved using SM with the PSP and BSP are 5.82% and 5.50% (when applied as SM(B2) and SM(B4)). There is again a lack of consistency however as revenue drops by -1.12% (using SM(B4)) and -26.32% (using SM(3)) with PSP and BSP respectively.

#### 5.4.2.3 iii) Experiment Three

Finally we undertake the same experiments using the third workload, which contains 3,461,612 requests to the the NASA Kennedy Space Center web-server in Florida. Results are shown in Table 5.3 and Figure 5.4. It is shown that the revenue of the system can be improved as much as 69.19% and 44.14% when BSP and PSP are applied to the system compared with the NSP; a further 45.91% and 0.76% improvement can be achieved when the SA and AR(1) are applied to the system along with BSP and PSP respectively (see Table 5.3).

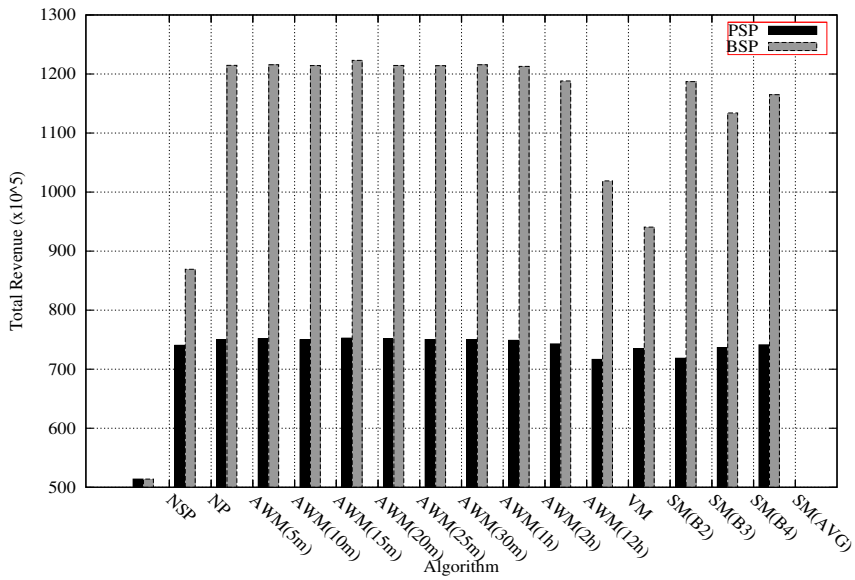


Figure 5.4: Revenue using Active Window Model (AWM), Voting Model (VM), and Selective Model (SM) over the third workload

The AWM performs the best compared with the other two models where the system revenue can be improved by 4.68% and 40.69% with the PSP and BSP policies, when AWM is applied every ten and twenty minutes respectively (the improvement is between 0.96%-1.63% and 29.33%-39.71% using the remaining categories of AWM with PSP and BSP respectively).

When VM and SM are applied with PSP, it does not provide a good improvement in system revenue (-3.19%) with VM and from 0.08% to -2.98% with SM. VM however gives a good improvement in system revenue with BSP where the improvement reaches 15.08%. SM also provide a reasonable improvement (from 8.21% with SM(B2) and up to 36.56% with SM(B3)) in system revenue when applied alongside BSP.

### 5.4.3 Analysis

Tables 5.4, 5.5 and 5.6 provide a useful summary of these findings. Essentially we contrast an enterprise system with fixed resources (NSP) with several alternatives: a system that employes a dynamic server switching policy (PSP or BSP); a system that uses PSP or BSP, and a *single* forecasting scheme; and finally a system that employes PSP or BSP, and a *meta-forecasting* scheme. There are some interesting observations from this data:

- Dynamic server switching (using PSP or BSP) improves revenue in all cases. The Bottleneck Aware Switching policy is particularly effective;
- Using a single forecasting scheme in tandem with PSP or BSP is difficult. First, no one scheme wins out across all workload (the best single policy includes AR(1), EMA, SMA, and SA over our three workloads). Second, if the wrong scheme is chosen, this may indeed reduce the overall revenue generated (it does so in more than half the cases we test);
- The meta-forecasting schemes always improve revenue when used in tandem with PSP or BSP. In the worst case the improved revenue will be negligible (1.63%, workload three, PSP, AWM(20m)); in the best case the revenue may be increased by around 40% (40.69% workload three, BSP, AWM(20m));
- The Active Window Model (AWM) proves to be the best scheme in all cases; on average this scheme gives an improvement in revenue of 15.1% over all three real-world workloads. The size of the active window is important and must therefore be subject to some pre-calculatation based on sample traces.

Table 5.1: Revenue gains for switching policy and forecasting combinations over the first workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 614.1 | 648.7 | 650 | 647.9 | 645.7 | 677.8 | 651.3 | 727.5 | 660.6 |
| Imp. over PSP (%) | - | 0 | 0.2 | -0.12 | -0.46 | 4.49 | 0.4 | 12.15 | 1.83 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 614.1 | 1249.5 | 1341.3 | 1333.7 | 1333.9 | 1360.6 | 1332.8 | 1346.5 | 1333.4 |
| Imp. over BSP (%) | - | 0 | 7.35 | 6.74 | 6.75 | 8.89 | 6.67 | 7.76 | 6.71 |

Table 5.2: Revenue gains for switching policy and forecasting combinations over the second workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 632.6 | 810.6 | 786.1 | 758.6 | 803.9 | 811 | 816.1 | 825 | 820.9 |
| Imp. over PSP (%) | - | 0 | -3.02 | -6.41 | -0.83 | 0.05 | 0.68 | 1.78 | 1.27 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 632.6 | 2200.1 | 2273.3 | 2306.2 | 2360.3 | 2309.2 | 2182.8 | 2155.5 | 2279.3 |
| Imp. over BSP (%) | - | 0 | 3.33 | 4.82 | 7.28 | 4.96 | -0.79 | -2.03 | 3.6 |

Table 5.3: Revenue gains for switching policy and forecasting combinations over the third workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 513.8 | 740.6 | 718.0 | 703.1 | 742.2 | 745.0 | 730.7 | 746.2 | 740.5 |
| Imp. over PSP (%) | - | 0 | -3.05 | -5.06 | 0.22 | 0.59 | -1.34 | 0.76 | -0.01 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue | 513.8 | 869.3 | 940.0 | 1268.4 | 1164.6 | 978.3 | 1260.7 | 1083.5 | 1206.3 |
| Imp. over BSP (%) | - | 0 | 8.13 | 45.91 | 33.97 | 12.54 | 45.02 | 24.64 | 38.77 |

Table 5.4: Analysis of the first workload

| Policy | NSP | PSP | Best Single Policy | Worst Single Policy | Best meta-policy |
|---|---|---|---|---|---|
| Total Revenue | 614.1 | 648.7 | (AR1) 727.5 | (SMA) 645.7 | (AWM(1h)) 739.9 |
| Imp. over PSP (%) | - | 0 | 12.15 | -0.46 | 14.06 |
| Policy | NSP | BSP | Best Single Policy | Worst Single Policy | Best meta-policy |
| Total Revenue | 614.1 | 1249.5 | (EMA) 1360.6 | (LPF) 1332.8 | (AWM(30m)) 1420.3 |
| Imp. over BSP (%) | - | 0 | 8.89 | 6.67 | 13.67 |

Table 5.5: Analysis of the second workload

| Policy | NSP | PSP | Best Single Policy | Worst Single Policy | Best meta-policy |
|---|---|---|---|---|---|
| Total Revenue | 632.6 | 810.6 | (AR1) 825 | (SA) 758.6 | (AWM(12h)) 909.8 |
| Imp. over PSP (%) | - | 0 | 1.78 | -6.41 | 12.24 |
| Policy | NSP | BSP | Best Single Policy | Worst Single Policy | Best meta-policy |
| Total Revenue | 632.6 | 2200.1 | (SMA) 2360.3 | (AR1) 2155.5 | (AWM(12h)) 2383.1 |
| Imp. over BSP (%) | - | 0 | 7.28 | -2.03 | 8.32 |

Table 5.6: Analysis of the third workload

| Policy | NSP | PSP | Best Single Policy | Worst Single Policy | Best meta-policy |
|---|---|---|---|---|---|
| Total Revenue | 513.8 | 740.6 | (AR1) 746.2 | (SA) 703.1 | (AWM(20m)) 752.7 |
| Imp. over PSP (%) | - | 0 | 0.76 | -5.06 | 1.63 |
| Policy | NSP | BSP | Best Single Policy | Worst Single Policy | Best meta-policy |
| Total Revenue | 513.8 | 869.3 | (SA) 1268.4 | (LO) 940.0 | (AWM(20m)) 1223 |
| Imp. over BSP (%) | - | 0 | 45.91 | 8.13 | 40.69 |

## 5.5 Summary

Through modelling and supporting simulation, we combine the *reactive* behaviour of two well known switching policies – the Proportional Switching Policy (PSP) and the Bottleneck Aware Switching policy (BSP) – with the *proactive* properties of several workload forecasting models. Seven forecasting models are used, including Last Observation, Simple Algorithm, Sample Moving Average, Low Pass Filter and Autoregressive Moving Average. As each of the forecasting schemes has its own bias, we also develop three meta-forecasting models (the Active Window Model, the Voting Model and the Selective Model) to ensure consistent and improved results.

We base our results on real-world workload traces from several sources, including from the San Diego Supercomputer Centre, from the ClarkNet Internet access provider for the Metro Baltimore-Washington DC area and, from the NASA Kennedy Space Center web-server in Florida. For each of the three real-world workloads, we contrast an enterprise system with fixed resources (no switching policy) with several alternatives: a system that employes a dynamic server switching policy (PSP or BSP); a system that uses PSP or BSP, and a *single* forecasting scheme; and finally a system that employes PSP or BSP, and a *meta-forecasting* scheme.

The results are significant in a number of respects: (i) Dynamic server switching (using PSP or BSP) improves revenue in all cases, the Bottleneck Aware Switching policy is particularly effective; (ii) Using a single forecasting scheme in tandem with PSP or BSP is difficult as no one scheme wins out across all workloads and, if the wrong scheme is chosen, this may lead to a reduction in the overall revenue generated by the system; (iii) The meta-forecasting schemes always improve revenue when used in tandem with PSP or BSP, in the worst case the improvement revenue will be negli-

gible, in the best case the revenue may be increased by around 40%; (iv) The Active Window Model (AWM) proves to be the best scheme in all cases, and on average this scheme gives an improvement in revenue of 15.1% over all three real-world workloads.

It has been found that the size of the active window plays crucial role and must therefore be subject to some pre-calculatation based on different approaches. The next chapter presents different techniques which can be applied in order to enhance the developed meta-forecasting models by monitoring the system and making the appropriate decisions when needed, based on the incoming requests. Also a historical prediction model, which exploits the periodicity of web traffic to predict workload will be applied to examine the effect of these developed models on system revenue.

# Chapter 6

# Dynamic Active Windows, Workload Pattern Analysis and Extreme Workloads

## 6.1 Introduction

In Chapter 5 we developed three new meta-forecasting models and investigated their effectiveness on the three real-world workloads employed in this thesis. We develop this work further in this chapter.

The Active Window Model seen in Chapter 5 was shown to produce encouraging results that is, improvements in system revenue were achieved. The improvements brought about by this scheme did however vary and are related to the size of the active window as well as the workload itself. We term this approach the use of *static active windows* and explore a new *dynamic active window* technique here. We base this new approach on two new metrics: (1) the burstiness factor (BF) and (2) the request arrival rate technique (ART).

In addition to this we explore whether it is possible employ pattern analysis to help in our forecasting of workload. Our assumption here is that internet services experience patterns of activity, for example they may be quiet at night and experience busy periods

mid-morning and afternoon, and in the early evening. We explore this idea through a historical prediction model.

Finally, we investigate which of the techniques developed in this thesis demonstrate robustness in periods of extreme activity. We ask whether dynamic server switching and forecasting are effective in managing flash crowds; this is done through the creation of synthetic workloads that mimic such activities.

### 6.1.1 Chapter Contributions

This chapter extends the work in the previous chapter; the following additional contributions are made:

- In the previous chapter we demonstrated the benefits of meta-forecasting models is overcoming the bias of one particular forecasting technique. In this chapter we extend the Active Window Model.

- In Chapter 5 we demonstrated that the active window model performs better than the other two meta-forecasting models (system revenue may improved by as much as 40%). We also served that the size of the active window plays curial role. Thus we develop a Dynamic Active Window Model (DAWM) to control the data sampling interval over which server switching decisions are made. The DAWM is based on two different techniques – the burstiness factor and the request arrival rate. This extended model shows that a further 51.5% improvement can be achieved when the switching server policy, meta-forecasting and dynamic active window management are employed together over a real-world workload based on Internet traces.

- We also introduce the notion that workloads demonstrate patterns of behaviour

over long periods. We employ Workload Pattern Analysis in our studies and investigate the application of prediction techniques based on this approach.

- A synthetic workload with extreme events, where the number of requests rise suddenly for several periods and then return the normal range is introduced in this chapter in order to study of the behaviour of the developed models in periods of extreme activities (e.g flash crowds).

- We show that request servicing capability can be improved by as much as 92% when the right combination of dynamic server switching and workload forecasting are used over real-world Internet traces and 103% over a workload with extreme events. We base our results on real-world workload traces from several sources and also on two different synthetic workloads generated with extreme fluctuations in workload.

## 6.2 Additional Related Work

Server workload is usually measured in terms of the incoming requests arrival rate (e.g. total number of operations or size of the files requested per unit of time). In this thesis the interval for collecting the data from the Internet workload has been chosen to be five minutes, giving a total of 288 intervals in a day. This is typical in the characterisation of data from websites ranging from retail industries to finance (see [52]).

In the previous chapter, three meta-forecasting models have been developed, including: the Active Window Model (AWM), the Voting Model (VM), and the Selective Model (SM) based on several different predictive algorithms. The active window model (AWM) proves to be the best scheme in all cases.

The work here is different from that seen in Chapter 5 as we use an approach where

the size of the active window changes dynamically based on the burstiness factor (BF) (as stated in [19]) as well as the request arrival rate technique (ART).

## 6.3 Predictive Models

Experiments in this chapter have been conducted over the workloads with five meta-forecasting models – the Active Window Model (AWM), the Dynamic Active Window Model (DAWM), the Voting Model (VM), the Selective Model (SM), and the Workload Pattern Analysis (WPA). The three forecasting models (AWM, VM, and SM) are documented in Chapter 5. The remaning models (DAWM and WPA) are described as follows:

### 6.3.1 Dynamic Active Window Model (DAWM)

In Chapter 5 we demonstrated that the Active Window Model (AWM) performs better than the other two meta-forecasting models (VM and SM). Added to this, the size of the active window plays crucial role and must therefore be subject to some pre-calculatation based on different approaches.

The main concern here is how to monitor the behaviour of the developed predictors in order to improve the quality of the developed model as well as to reduce the algorithm overhead. The requests' arrival rate parameter is one of the different systems parameter that can be monitored [19]. The monitoring process of the requests' arrival rate can be done in three different scenarios; i) each time a request arrives at the front-end of the system, ii) at fixed times by using static time slot scheduling, iii) at non-fixed times by using dynamic time slot scheduling. It is clear that the first option, where the monitoring of the system is conducted each time a new request arrives is naive, not cost free, and requires large calculation. While the second scenario is

conducted at pre-fixed slots of time that are already determined at the design stage. The previous chapter (Chapter 5) is based on this scenario. The main concern of this chapter is to develop the third scenario where the monitoring of the proposed system is done at non-fixed times and the size of the time slots changes dynamically based on the requests arrival rate. Thus the model (*DAWM*) is developed where the duration of the active window's size for collecting the data points for all predictors is variable as it is very difficult to set a duration interval that fits with all possible requests arrival rates due to its heavy tailed pattern. The active window size is calculated based on either a burstiness factor – the window size decreases when the workload becomes more bursty – or based on the correlation between the request's arrival rate and the mean arrival rate for incoming requests to the system.

#### 6.3.1.1 i) Burstiness Technique and Monitoring Window Size

Due to the significant variance of the Internet traces, and in order to produce a system with good performance, especially when the workload demand is high, burstiness needs to be controlled and studied [18].

The burstiness factor – Penalisation Included (PI) – which is documented in [19], is applied in this work. The ($DAWM_{PI}$) can be computed by comparing the mean arrival rate of HTTP transactions during all slots ($\mu$) with the arrival rate of the current slot ($\lambda$).

The current slot is considered to be a bursty slot If the arrival rate of the current slot is greater than the mean arrival rate of HTTP transactions during all slots ($\lambda > \mu$). That is, the burstiness factor is defined as the relation between the cumulative number of the whole slots that satisfy ($\lambda > \mu$) which is called ($s'$) and the current number of slots ($s$).

In other words, the burstiness factor is then computed by dividing the number of bursty slots ($s'$) by the number of slots ($s$) and then multiplying the result with $(1+\alpha)$; where ($\alpha$) is used to capture continued increases in traffic volume. It is considered that a maximum of $j$ consecutive bursty slots may lead to a proportional raise in the burstiness factor, where $\alpha = (0.1 * j), for j \in 1, ..., 10$. The maximum record of 10 slots has been chosen as in [19]; this is essentially saying that if the burstiness detected in the arrival rate is extreme, then the burstiness factor will be doubled every 10 slots.

$$b(s) = \frac{s'}{s} \times (1+\alpha). \tag{6.1}$$

Where the whole slots in the system is represented by ($s'$) and the ($s$) represents the bursty slots. Note that the range of this factor is grouped between 0 to 1 to avoid congestion [19].

After computing the burstiness factor, the duration of the next slot $d(s+1)$ is computed based on the burstiness factor of the current slot $b(s)$ and the previous slot $b(s-1)$, as follows;

$$d(s+1) = \begin{cases} \dfrac{d(s) \times \Delta}{1 + b(s) + b(s-1)}, b(s) \geq b(s-1) \\ \\ \dfrac{d(s) \times \Delta}{1 + b(s) - b(s-1)}, b(s) < b(s-1) \end{cases} \tag{6.2}$$

By calculating the next check-point based on the burstiness factor of the previous two slots, the length to the next check-point increases if a decreased burstiness is perceived, and is brought sooner if the burstiness increases. Therefore, a sudden reduction and enlargement of the burstiness can be forecasted [19].

#### 6.3.1.2 ii) Arrival Rate Technique

The second approach that used in order to monitor the proposed model is known as the arrival rate technique ($DAWM_{ART}$). The ($DWAM_{ART}$) is developed in this research based on the arrival rate of the current request $\lambda_r$, compared with the mean arrival rate of the previous incoming requests to the system $\mu_r$ where;

$$\mu_r = \frac{\sum_{x=1}^{r} \lambda_r}{r}. \tag{6.3}$$

The data points from different predictors are collected (the check point) when the arrival rate of the current request $\lambda$ is greater than the mean arrival rate of requests $\mu$.

### 6.3.2 Workload Pattern Analysis (WPA)

We finally introduce a historical prediction model, which exploits the periodicity of web traffic to predict workload. In this model the predicted number of requests is related to the previous number of requests found at the same time of day (the number of requests at midnight for a specific day is related to the number of requests at midnight recorded during the previous week).

As an example of using the new developed technique consider the following; suppose that the number of requests at Monday midnight was equal to 150 requests and also it has been found that the number of requests is 200 requests at the Tuesday midnight. In this case and in order to compute the number of requests at Wednesday midnight, the number of requests at Monday and Tuesday are used as follow; (No. of requests at Wednesday midnight = (No. of requests at Monday midnight + No. of requests at Tuesday midnight) / 2). In this case the number of requests at Wednesday midnight can be computed by adding the values of 200 and 150 and dividing them by

2, (200+150) / 2 = 175.

## 6.4 Experimental Results and Analysis

We have developed a supporting simulator to allow us to verify the behaviour of our theoretical models. We prime the simulator with measured values from an in-house test platform, or using values from supporting literature where these are not attainable – the details of the experiment setup are documented in Chapter 3.

In the first experiment, the results from applying the new meta-forecasting model – Dynamic Active Window Model (DAWM) – compared with the previous forecasting model – Active Window Model (see Section 5.4.2.1 in Chapter 5) – along with dynamic server switching to three real-world Internet traces are provided. The results from applying the periodic forecasting model Workload Pattern Analysis (WPA) over the three real-world Internet traces is found in the second experiment. The third experiment repeats the same analysis of the different models (AWM, DAWM, VM, SM, and WPA) but on a synthetic workload containing extreme events.

### 6.4.1 Experiment One

The first experiment is conducted using the new adaptive forecasting model – Dynamic Active Window Model (DAWM) – which is developed based on the previous meta-forecasting model (AWM), as it has been found that AWM is more effective than the other two meta models from Chapter 5. The DAWM is applied over the real-world workloads seen previously.

The results from the DAWM compared with the AWM based on the three workload and using PSP and BSP are shown in Tables 6.1 and 6.2 respectively.

Table 6.1 shows that using the DAWM either using the burstiness factor ($DAWM_{PI}$)

or arrival rate technique ($DAWM_{ART}$) performing along side PSP is better (more revenue) than that from the original PSP without prediction; with just one exception when the DAWM is applied using the burstiness factor over the second workload where the revenue dropped by -0.3%. Table 6.1 shows also that when the AWM is applied every one hour, the improvement is improved by at least 14.06% and this is the best improvement can be obtained from applying the AWM over the first workload. A further 5.44% improvement is achieved when the $DAWM_{ART}$ is used along side PSP over the same workload.

Applying the new developed meta-forecasting (DAWM) over the second workload does not improve the system's performance as much as the AWM does, as the best improvement that can be obtained from applying the DAWM is 1.6% ($DAWM_{ART}$, 1.6%). While the revenue can be improved by 12.24% when the $AWM_{12h}$ is applied.

The performance of the system can improve by up to 63.8% from applying the $DAWM_{PI}$ with PSP over the third workload, while the best improvement from applying the AWM is obtained when it is applied every 20 minutes (1.63%).

On average, using the best AWM alongside PSP over the three workloads can improve the system revenue by 9.31%, while at least 25.14% and 7.43% improvement are achieved from applying the DAWM using the burstiness factor ($DAWM_{PI}$) and arrival rate technique ($DAWM_{ART}$) respectively.

Table 6.2 shows the results from applying the DAWM using both the burstiness factor and the arrival rate technique with BSP over the three workloads, it also shows the best and worst AWM compared with that obtained from applying the BSP with no prediction.

It can be seen from the table that applying the DAWM always performs better than applying the switching policy without forecasting. DAWM does not improve the

system performance over the first workload compared with applying BSP along side the best active window model ($AWM_{30m}$). It performs however better than using BSP without prediction as the improvements are 9.4% and 8.8% respectively.

The results from Table 6.2 show also at least 0.5% and at most 8.3% improvement in system revenue when the worst and best AWM is applied compared to that from the original BSP without prediction over the second workload. On the other hand, the revenue from applying the DAWM using the burstiness factor ($DAWM_{PI}$) on the given workload can be up by 91.5% compared with BSP without prediction. Added to this, the system performance can be improved by 50.4% when the $DAWM_{PI}$ is applied with BSP over the third workload, while the largest improvement can be obtained from applying AWM on the given workload is 40.7% with $AWM_{20m}$.

The performance from applying the $DAWM_{ART}$ over the second and third workloads are 4.3% and 37.3% respectively.

The behaviour of the best AWM over the three given workloads on average is 20.9% while, the average improvement from applying the $DAWM_{PI}$ is around 50.43%. The average improvement from applying the $DAWM_{ART}$ over the three workloads is 16.8%.

### 6.4.2 Experiment Two

The second experiment is conducted over the workload with extreme events (see Figure 6.1) using the four meta-forecasting models (AWM, DAWM, VM, and SM).

The main reason behind using this synthetic workload that includes extreme fluctuations is to study the effect of the different meta-forecasting models. As stated before, most e-Business applications are subject to enormous variations in workload demand [10]; in addition, the traffic to such sites can become three or four times greater than

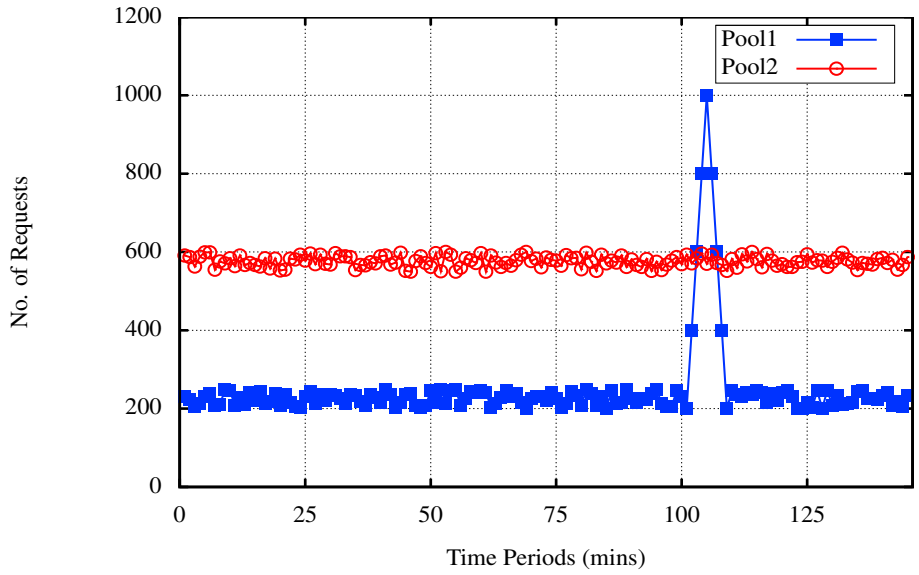


Figure 6.1: A sample of the total requests in the synthetic workload for both application pools

the average traffic (for example) and the server capacity fails in serving active customers [41]. Our workload is purely illustrative in that the time in which the traffic spike occurs, or its extent, is chosen randomly.

The generated workload has a normal activity in one pool while there is a sudden change in the second pool, which retains normal activity after the sudden change (see Figure 6.1).

The results from applying the meta-forecasting models over the synthetic workload with extreme event are shown in Table 6.3 and Figure 6.2.

These results are interesting in that the PSP does not react well to these events – PSP decreases revenue over a system which employs no switching policy (see Table 6.3). In contrast, BSP does provide an improvement in system revenue, albeit reduced when the additional predictive algorithms are employed.

The results of the meta-forecasting schemes (Figure 6.2) again show that PSP is

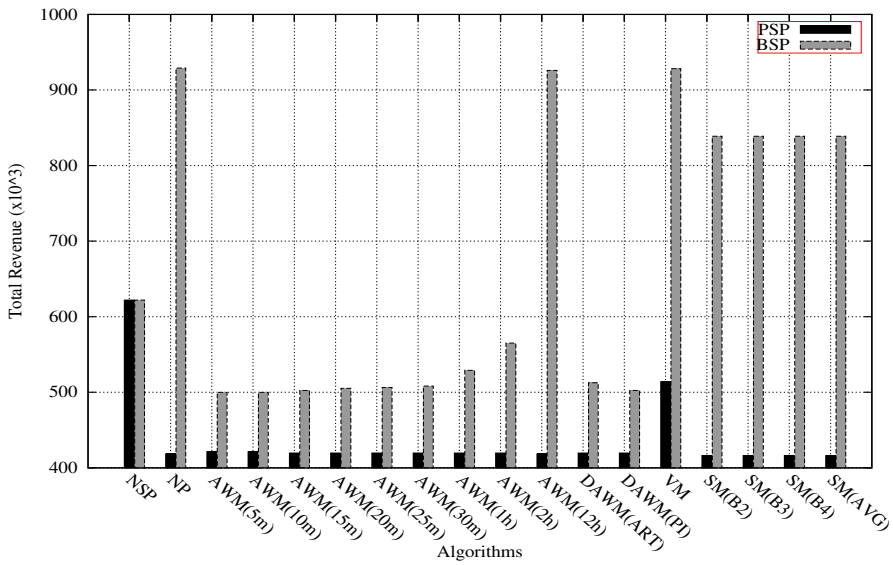


Figure 6.2: Revenue using Active Window Model (AWM), Dynamic Active Window Model (DAWM), Voting Model (VM), and Selective Model (SM) over the fourth workload

not effective. BSP is again the scheme of choice, however some interesting observations can be made. BSP together with the AWM is extremely sensitive to the extreme events. If the active window is too small then the extreme events will effect the revenue generating capabilities of the system (as to be expected); as the active window size increases, so the scheme is more able to cope with these extreme events (the same thing can be said to the DAWM using the burstiness factor ($DAWM_{PI}$) or arrival rate technique ($DAWM_{ART}$)). VM and SM also provide robust results in response to extreme events, although neither scheme improves on the application of BSP alone.

It is clear that when AWM is applied every hour ($AWM_{1h}$) over the first workload, it performs better than DAWM with the arrival rate technique ($DAWM_{ART}$). This is due to the behaviour of the workload (Figure 6.3). To understand this we consider the characteristics of the two workloads: If we look at the number of requests for each of the application pools, we find that the workloads for the two pools cross only a small

number of times (in particular at the beginning of the workload) and the differentiation between the number of requests is small. Thus applying DAWM with the arrival rate technique (the server switching process is performed between the two pools when the arrival rate of the workload is greater than the mean arrival rate of the workload) is an effective approach as the decision process is performed constantly. Contrast this with the case when AWM is applied over this workload, where the decision process (of whether to switch or not) is applied less frequently (this is due to the behaviour of this technique) where some of the more subtle servers switching decisions are not applied.

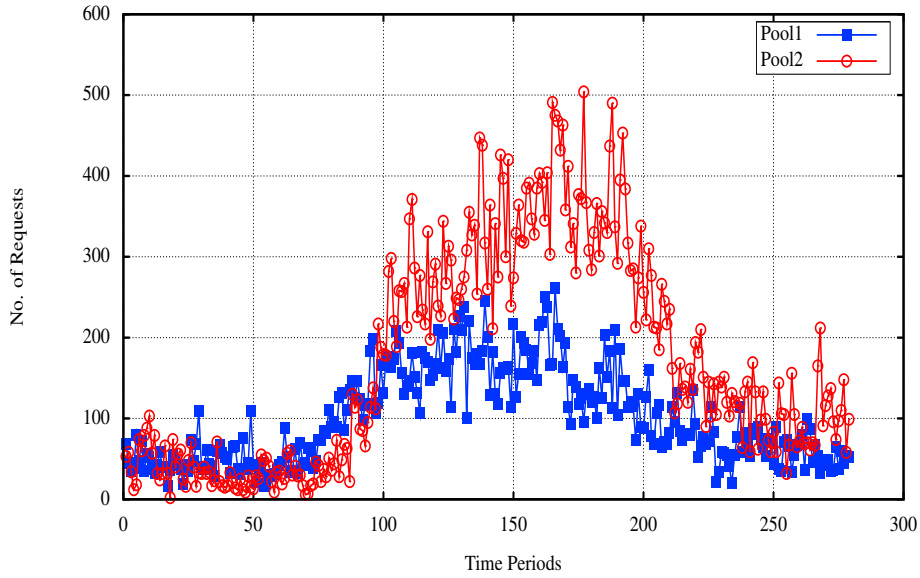


Figure 6.3: The total requests in the first real-world workload for both application pools

The characteristics of the second workload are quite different (Figure 6.4). It is clear that the workloads cross constantly and the load between the two pools oscillates. Added to this, the differentiation of the number of requests in the two application pools are larger than in the first workload. Therefore, we see an interesting interaction be-

tween the active window size and the point at which decisions are made as to whether the server switching process should be conducted or not: When AWM is applied every twelve hours, this occurs just before the workloads of both pools cross each other (which is the best time to switch the severs between the pools). This means therefore that applying AWM every twelve-hours is testing the switching of the servers at the best time in relation to this workload, and because of this it is performing better compared with the other models. Although the number of switched severs between the two pools is large when DAWM is applied over the second workload, we must remember that the switching server process is not cost free and this may affect the total revenue of the proposed application.

This analysis highlights the impact of the workload on the choice of server switching technique (and the revenue gained), and highlights the need to be able to predict workload to some degree. This topic is addressed later in the thesis.

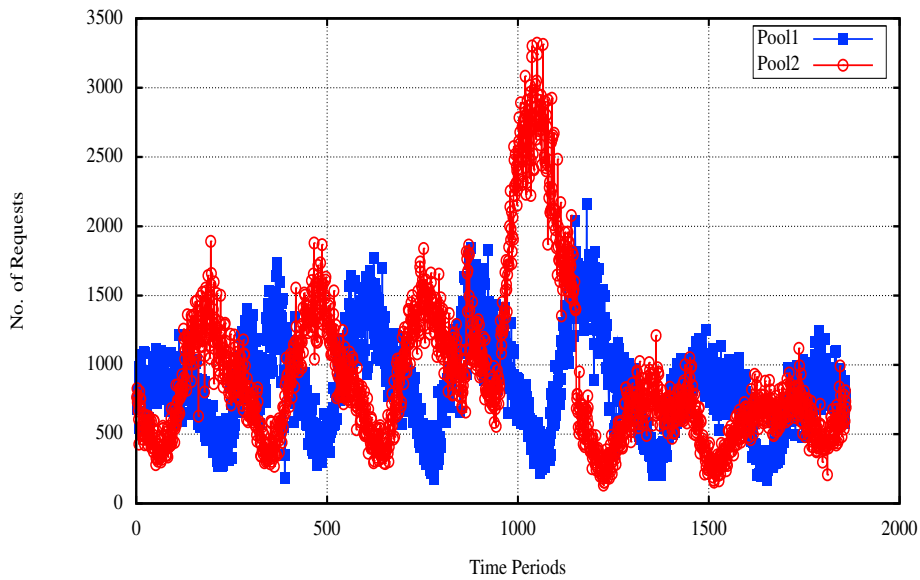


Figure 6.4: The total requests in the second real-world workload for both application pools

### 6.4.3 Experiment Three

The final experiment has been undertaken using the fourth developed model – Workload Pattern Analysis (WPA) – and is applied over the four workloads (three real-world Internet traces and the synthetic workload with extreme events). Results from applying WPA are compared with those from using no switching policy (NSP), vanilla PSP or BSP, and PSP or BSP with the best meta-forecasting scheme (AWM, DAWM, VM, and SM); see Figure 6.5.

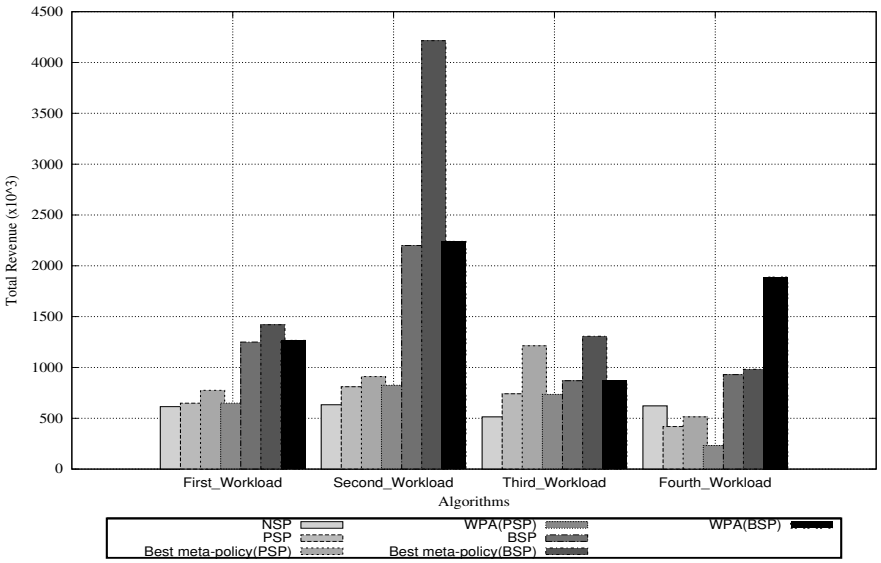


Figure 6.5: Revenue using Workload Pattern Analysis (WPA) under the four workloads

The new WPA scheme also performs effectively, although is less reliable than dynamic server switching with meta-forecasting (columns 4 and 7). It is this fact which we wish to highlight in the results – WPA is clearly an effective scheme, and in the case of workload four (with extreme events) produces some surprisingly good results; however, this suggests that PSP and BSP are not well suited for handling extreme events and there are improvements to be made in this regard. If one is seeking consistent results, then dynamic server switching with meta-forecasting is still a good choice.

### 6.4.4 Analysis

Tables 6.4 and 6.5 provide a useful summary of these findings. Essentially we contrast an enterprise system with fixed resources (NSP) with several alternatives: a system that employes a dynamic server switching policy (PSP or BSP); a system that uses PSP or BSP, and a *single* forecasting scheme; and finally a system that employes PSP or BSP, and a *meta-forecasting* scheme including DAWM and WPA. There are some interesting observations from this data.

- The meta-forecasting schemes always improve revenue when used in tandem with PSP or BSP over the three real-world Internet traces. In the worst case the improved revenue will be negligible (12.24%, workload two, PSP, $AWM_{(12h)}$); in the best case the revenue may be increased by around 92% (91.5%, BSP, $DAWM_{(PI)}$, workload two);

- Applying the meta-forecasting schemes over the synthetic workload with extreme event also improve the system revenue by 23% (22.8%, PSP, VM) and 103% (103.3%, BSP, WPA) with PSP and BSP respectively. We believe that dealing with extreme events is more complex and requires further investigation to ensure consistent results;

- On average these meta-forecasting schemes give an improvement in the system revenue of 29.59% and 64.72% over all the four workloads.

Table 6.1: Analysis of applying the forecasting models (DAWM) and (AWM) alongside PSP over the three real-world workloads

| Policy | NSP | PSP | Worst AWM | Best AWM | $DAWM_{PI}$ | $DAWM_{ART}$ |
|---|---|---|---|---|---|---|
| The First Workload (Revenue x $10^3$) | | | | | | |
| Revenue | 614.1 | 648.7 | ($AWM_{12h}$) 648.7 | ($AWM_{1h}$) 739.9 | 726.1 | 775.2 |
| Imp. (%) | - | 0 | 0 | 14.06 | 11.93 | 19.50 |
| The Second Workload (Revenue x $10^4$) | | | | | | |
| Revenue | 632.6 | 810.6 | ($AWM_{5m}$) 814.9 | ($AWM_{12h}$) 909.8 | 807.9 | 823.4 |
| Imp. (%) | - | 0 | 0.5 | 12.24 | -0.3 | 1.6 |
| The Third Workload (Revenue x $10^5$) | | | | | | |
| Revenue | 513.8 | 740.6 | ($AWM_{12h}$) 742.8 | ($AWM_{20m}$) 752.7 | 1213.3 | 749.2 |
| Imp. (%) | - | 0 | 0.3 | 1.63 | 63.8 | 1.2 |

Table 6.2: Analysis of applying the forecasting models (DAWM) and (AWM) alongside BSP over the three real-world workloads

| Policy | NSP | BSP | Worst AWM | Best AWM | $DAWM_{PI}$ | $DAWM_{ART}$ |
|---|---|---|---|---|---|---|
| The First Workload (Revenue x $10^3$) | | | | | | |
| Revenue | 614.1 | 1249.5 | ($AWM_{2h}$) 971.9 | ($AWM_{30m}$) 1420.3 | 1366.8 | 1360.1 |
| Imp. (%) | - | 0 | -22.2 | 13.7 | 9.4 | 8.8 |
| The Second Workload (Revenue x $10^4$) | | | | | | |
| Revenue | 632.6 | 2200.1 | ($AWM_{10m}$) 2211.3 | ($AWM_{12h}$) 2383.1 | 4214.4 | 2294.4 |
| Imp. (%) | - | 0 | 0.5 | 8.3 | 91.5 | 4.3 |
| The Third Workload (Revenue x $10^5$) | | | | | | |
| Revenue | 513.8 | 869.3 | ($AWM_{12h}$) 1188.3 | ($AWM_{20m}$) 1223.0 | 1307.1 | 1193.3 |
| Imp. (%) | - | 0 | 36.70 | 40.7 | 50.4 | 37.3 |

Table 6.3: Revenue gains for the switching policy and forecasting combinations over the fourth workload

| | | | PSP + Predictive Algorithm | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | NSP | PSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue (x$10^3$) | 622 | 418.5 | 416.2 | 416.2 | 416.2 | 416.2 | 416.2 | 416.2 | 416.2 |
| Imp. over PSP (%) | - | 0 | -0.56 | -0.56 | -0.56 | -0.56 | -0.56 | -0.56 | -0.56 |
| | | | BSP + Predictive Algorithm | | | | | | |
| Policy | NSP | BSP | LO | SA | SMA | EMA | LPF | AR(1) | AR(2) |
| Total Revenue (x$10^3$) | 622 | 929.1 | 839.4 | 839.4 | 838.9 | 838.9 | 839.4 | 581.3 | 581.3 |
| Imp. over BSP (%) | - | 0 | -9.65 | -9.65 | -9.71 | -9.71 | -9.65 | -37.4 | -37.4 |

Table 6.4: Analysis of applying the forecasting model alongside PSP over the four workloads

| Policy | NSP | PSP | Best Single Policy | Worst Single Policy | Best meta-policy |
|---|---|---|---|---|---|
| The First Workload (Revenue x $10^3$) | | | | | |
| Revenue | 614.1 | 648.7 | (AR1) 727.5 | (SMA) 645.7 | ($DAWM_{(ART)}$) 775.2 |
| Imp. (%) | - | 0 | 12.15 | -0.46 | 19.50 |
| The Second Workload (Revenue x $10^4$) | | | | | |
| Revenue | 632.6 | 810.6 | (AR1) 825 | (SA) 758.6 | ($AWM_{(12h)}$) 909.8 |
| Imp. (%) | - | 0 | 1.78 | -6.41 | 12.24 |
| The Third Workload (Revenue x $10^5$) | | | | | |
| Revenue | 513.8 | 740.6 | (AR1) 746.2 | (SA) 703.1 | ($DAWM_{(PI)}$) 1213.3 |
| Imp. (%) | - | 0 | 0.76 | -5.06 | 63.80 |
| The Fourth Workload (Revenue x $10^3$) | | | | | |
| Revenue | 622 | 418.5 | (All Predictors) 416.2 | (All Predictors) 416.2 | (VM) 513.9 |
| Imp. (%) | - | 0 | -0.56 | -0.56 | 22.80 |

Table 6.5: Analysis of applying the forecasting model alongside BSP over the four workloads

| Policy | NSP | BSP | Best Single Policy | Worst Single Policy | Best meta-policy |
|---|---|---|---|---|---|
| The First Workload (Revenue x $10^3$) | | | | | |
| Revenue | 614.1 | 1249.5 | (EMA) 1360 | (LPF) 1332.8 | ($AWM_{(30m)}$) 1420.3 |
| Imp. (%) | - | 0 | 8.84 | 6.67 | 13.67 |
| The Second Workload (Revenue x $10^4$) | | | | | |
| Revenue | 632.6 | 2200.1 | (SMA) 2360.3 | (AR1) 2155.5 | ($DAWM_{(PI)}$) 4214.4 |
| Imp. (%) | - | 0 | 7.28 | -2.03 | 91.5 |
| The Third Workload (Revenue x $10^5$) | | | | | |
| Revenue | 513.8 | 869.3 | (SA) 1268.4 | (LO) 940.0 | ($DAWM_{(50.4)}$) 1307.1 |
| Imp. (%) | - | 0 | 45.91 | 8.13 | 50.4 |
| The Fourth Workload (Revenue x $10^3$) | | | | | |
| Revenue | 622 | 929.1 | (LO, SA, and LPF) 839.4 | (AR1 & AR2) 581.3 | (WPA) 1888.9 |
| Imp. (%) | - | 0 | -9.71 | -37.4 | 103.30 |

## 6.5 Summary

Through modelling and supporting simulation, we combine the *reactive* behaviour of two well known switching policies – the Proportional Switching Policy (PSP) and the Bottleneck Aware Switching policy (BSP) – with the *proactive* properties of several workload forecasting models – the Active Window Model (AWM), the Dynamic Active Window Model (DAWM), the Voting Model (VM), the Selective Model (SM), and the Workload Pattern Analysis (WPA). Each combination is evaluated for its effectiveness against both real-world workloads and a synthetic workload with extreme events.

The results are significant in a number of respects: (i) The data points are collected from these predictors within non-fixed periods (Active Windows) to ensure consistent and improved results from several predictors and the active window sizes are computed based on two techniques, the burstiness factor (BF) and request arrival rate technique (ART). The result from applying the burstiness factor techniques show that the revenue can be improved by up to 64% and 92% when the burstiness factors technique is applied over the real-world workload alongside PSP and BSP respectively (63.8%, PSP, $DAWM_{(PI)}$, workload three) and (91.5%, BSP, $DAWM_{(PI)}$, workload two). (ii) Applying dynamic server switching and prediction is less effective for workload containing extreme events. We have shown that an improvement is possible (22.80% and 103.30%) when the VM and WPA are applied with PSP and BSP. This research however remains the subject of future work.

# Chapter 7

# Conclusions and Future Research

E-Business applications are subject to significant variations in workload and this can cause exceptionally long response times for users, the timing out of client requests and/or the dropping of connections. One solution is to host these applications in virtualised server pools, and to dynamically reassign compute servers between pools to meet the demands on the hosted applications. Switching servers between pools is not without cost, and this must therefore be weighed against possible system gain.

This work is concerned with dynamic resource allocation for multi-tiered, cluster-based web hosting environments. Dynamic resource allocation is reactive, that is, when overloading occurs in one resource pool, servers are moved from another (quieter) pool to meet this demand. Switching servers comes with some overhead, so it is important to weigh up the costs of the switch against possible system gains. In this thesis we combine the *reactive* behaviour of two server switching policies – the Proportional Switching Policy (PSP) and the Bottleneck Aware Switching Policy (BSP) – with the *proactive* properties of several workload forecasting models.

The principal contributions of this thesis are as follows:

- First, we study the impact of *server switching time* in distributed and dynamic

enterprise systems. The switching time is defined as the time taken to reallocate servers between applications. Our aim is to investigate the link between switching time and total system performance, as well as how the switching policies themselves behave with changeable switching times. For this purpose, we integrate two well known switching policies – the Proportional Switching Policy (PSP) and the Bottleneck-aware Switching Policy (BSP) – with variable switching times in a test system. Experiments are conducted on synthetic workloads and also on three real-world Internet traces containing two days, two weeks, and two months worth of HTTP requests. It has been found that the reduction in revenue due to increased reallocation intervals generally holds over all server switching policies, with some exceptions when the experiments are conducted alongside the synthetic workload. While when the real-world Internet traces is used the results show that there are no any relationship between the number of server switches and the system revenue.

- In our second contribution we extended the concept of *reactive* dynamic server switching – where changes in demand on a system would trigger the reallocation of servers to applications – with several forecasting schemes that would allow the server switching to become *proactive* – that is, a server switch would be initiated before the demand on the system changed. A multi-tiered, cluster-based, multi-server solution is modelled, which provides bottleneck identification through the use of convex polytopes and also employes admission control. A workload model is also constructed from the characterisation of real data from several different sources. Several schemes for workload prediction have been introduced including: Last Observation (LO), Simple Algorithm (SA), Sample

Moving Average (SMA), Exponential Moving Average (EMA), Low Pass Filter (LPF), and Autoregressive Integrated Moving Average (ARIMA). A comparison between the forecast accuracy of these schemes *in combination with dynamic server switching* is conducted using several metrics – Mean Square Error (MSE), Mean Average Percentage Error (MAPE), Mean Absolute Deviation (MAD) and the Cumulative sum of Forecast Error (CFE). It has been shown that the system revenue can be improved by as much as 46% when the dynamic server-switching is combined correctly with the workload forecasting. However the accuracy of each predictor is different from one policy to another and there is no general case where improvements in revenue can be guaranteed.

- Third, we extend our models in order to address issues that arise when a single forecasting model is used, as each of the forecasting schemes is shown to have its own bias. As a result, three different meta-forecasting algorithms are developed – Active Window Model (AWM), Voting Model (VM), and Selective Model (SM). In the first model (AWM), data points are collected during an active window, and a predictor which delivers the best revenue for the last active window, is employed. All predictors are used in the voting model (VM) and a server switch is enacted if the majority vote requires. The selective model (SM) applies the best predictor from the last time period to the next time period. It has found that the Active Window Model (AWM) is the best scheme in all cases; on average this scheme gives an improvement in revenue of 15.1% over all three used real-world Internet traces. The size of the active window is therefore important and must be subject to some calculation using different policies.

- Finally, we further extend the Active Window Model (AWM) to a Dynamic Ac-

tive Window Model (DAWM), where the size of the active window for collecting data points for all predictors is not constant. The window size is calculated based on either a burstiness factor the window size decreases when the workload becomes more bursty or based on the correlation between the requests arrival rate and the mean arrival rate for incoming requests to the system. In addition to this, we introduce a historical prediction model the Workload Pattern Analysis (WPA) model which exploits the periodicity of web traffic to predict workload where the predicted number of requests in this model is related to the previous number of requests found at the same time of day (e.g. the number of requests at mid-night for a specific day is related to the number of requests at midnight recorded during the previous week). All schemes are tested on real-world workloads and also workloads containing extreme events. The obtained results from applying the new developed techniques show that the revenue can be improved by up to 64% and 92% when it is applied over the real-world workload alongside PSP and BSP respectively. Added to this, it has been found that the Workload Pattern Analysis (WPA) model is able to bring about improvements to the switching server policy (BSP), revenue is up 103.30% over the workload containing extreme events.

This work has been published in numerous papers, and we are grateful to the reviewers of this work for their helpful and constructive feedback. We recognise that this work has several limitations, most notably that fault tolerance is not considered and that server switching as described in this thesis means *between server pools in application tiers*. Clearly this work can be extended in this regard, although we note that techniques described here have been applied to industry-based systems.

## 7.1 Further Work

We make several recommendations in relation to future research:

- To extend the work to server switching between tiers – that is, server switching need not be restricted to just between the application tiers in our three-tier architecture;
- To develop this work in the context of fault tolerance. Adam Chester has touched on this in his own research and we believe that this represents a useful starting point to research in this area;
- This work has enormous potential in the context of power management. Despite significant interest in this, this fell outside the domain of this thesis. It is possible that the metric *system revenue* could be replaced by some energy metric. Researchers at the University of Newcastle have looked into this topic although it remains an area which requires in depth analysis;
- Although we have employed two dynamic server switching techniques in this work (BSP and PSP), several other alternatives are possible. We welcome further study in this area and believe that there remains opportunity to advance this area.
- The system model that is developed in this thesis is based on switching the available servers between two different applications; schemes that investigate more than two applications may of course be developed in order to generalise the proposed results - we leave this as the subject of future work.

# References

[1] Arlitt, M. and Jin, T. (2000). A Workload Characterization Study of the 1998 World Cup Web Site. *IEEE Network*, **14**(3), 30–37. 34

[2] Arlitt, M. and Williamson, C. (1996). Web server workload characterization: the search for invariants. *SIGMETRICS Perform. Eval. Rev.*, **24**(1), 126–137. 45, 57, 67

[3] Balbo, G. and Serazzi, G. (1997). Asymptotic Analysis of Multiclass Closed Queueing Networks: Multiple Bottlenecks. *Performance Evaluation*, **30**(3), 115–152. 22

[4] Barford, P. and Crovella, M. (1998). Generating Representative Web Workloads for Network and Server Performance Evaluation. *SIGMETRICS Performance Evaluation Review*, **26**(1), 151–160. 34

[5] Bolch, G., Greiner, S., deMeer, H., and Trivedi, K. (2006). *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. Wiley-Blackwell; 2nd Edition edition, New York, NY, USA. 31, 38, 40

[6] Buzen, J. (1973). Computational algorithms for closed queueing networks with exponential servers. *Commun. ACM*, **16**, 527–531. 21, 31

[7] Calzarossa, M. and Serazzi, G. (1993). Workload characterization: a survey. *Proceedings of the IEEE*, **81**(8), 1136 –1150. 65, 67

[8] Casale, G. and Serazzi, G. (2004). Bottlenecks identification in multiclass queueing networks using convex polytopes. *In 12th Annual Meeting of the IEEE Int'l Symposium on Modelling, Analysis, and Simulation of Comp. and Telecommunication Systems (MASCOTS)*. xvii, 22, 23, 24, 37

[9] Cavendish, D., Koide, H., Oie, Y., and Gerla, M. (2010). A mean value analysis approach to transaction performance evaluation of multi-server systems. *Concurr. Comput. : Pract. Exper.*, **22**(10), 1267–1285. 21, 32

[10] Chester, A. P., Xue, J. W. J., He, L., and Jarvis, S. A. (2008). A system for dynamic server allocation in application server clusters. In *ISPA '08: Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pages 130–139, Washington, DC, USA. IEEE Computer Society. xvii, 9, 11, 24, 118

[11] Contreras, J., Espinola, R., Nogales, F. J., and Conejo, A. J. (2003). Arima models to predict next-day electricity prices. *Power Systems, IEEE Transactions on*, **18**(3), 1014 – 1020. 71

[12] Cuomo, G. (2000). *IBM WebSphere Application Server Standard and Advanced Editions; A methodology for performance tuning*. IBM. 25

[13] Dattatreya, G. R. (2008). *Performance Analysis of Queuing and Computer Networks (Chapman & Hall/Crc Computer & Information Science Series)*. Chapman & Hall/CRC. 18, 19

[14] Dushay, N., French, J. C., and Lagoze, C. (1999). Predicting indexer performance in a distributed digital library. In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '99, pages 142–166, London, UK, UK. Springer-Verlag. 70

[15] Eager, D. and Sevcik, K. (1986). Bound hierarchies for multiple-class queuing networks. *J. ACM*, **33**, 179–206. 22

[16] Faraz, A. and Vijaykumar, T. (2010). Joint optimization of idle and cooling power in data centers while maintaining response time. *SIGPLAN Not.*, **45**(3), 243–256. 25

[17] Federgruen, A. and Groenevelt, H. (1986). The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality. *Oper. Res.*, **34**(6), 909–918. 16

[18] Gilly, K., Alcaraz, S., Juiz, C., and Puigjaner, R. (2004). Comparison of predictive techniques in cluster-based network servers with resource allocation. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, pages 545–552. 46, 91, 113

[19] Gilly, K., Alcaraz, S., Juiz, C., and Puigjaner, R. (2009). Analysis of burstiness monitoring and detection in an adaptive web system. *Computer Networks*, pages 668–679. 46, 112, 113, 114

[20] Haring, G. and Kotsis, G. (1995). Workload modeling for parallel processing systems. In *Proceedings of the 3rd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '95, pages 8–12, Washington, DC, USA. IEEE Computer Society. 65, 67

[21] He, L., Xue, J. W. J., and Jarvis, S. A. (2007). Partition-based profit optimisation for multi-class requests in clusters of servers. In *ICEBE '07: Proceedings of the IEEE International Conference on e-Business Engineering*, pages 131–138, Washington, DC, USA. IEEE Computer Society. 36

[22] Huberman, B. A. and Clearwater, S. H. (2005, http://www.hpl.hp.com/research/idl/papers/swings/). Swing options : A mechanism for pricing it peak demand. *Proceedings of 11th International Conference on Computing in Economics*. 22

[23] Keung, H., Dyson, J. R. D., Jarvis, S. A., and Nudd, G. R. (2003). Predicting the performance of globus monitoring and discovery service (mds-2) queries. In *Proceedings of the 4th International Workshop on Grid Computing*, GRID '03, pages 176–, Washington, DC, USA. IEEE Computer Society. 65, 69

[24] Krishna, K. (1993). *Book review: Introduction to Computer System Performance Evaluation by Krishna Kant (McGraw-Hill, 1992)*, volume 21. ACM, New York, NY, USA. 18

[25] Lam, S. (1983). A simple derivation of the mva and lbanc algorithms from the convolution algorithm. *IEEE Transactions on Computers*, **C-32**(11), 1062 –1064. 21

[26] Lazowska, E., Zahorjan, J., Graham, G., and Sevcik, K. (1984). *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 17, 18

[27] LBNL (2008). Internet Traffic Archive Hosted at Lawrence Berkeley National Laboratory. http://ita.ee.lbl.gov/html/traces.html. 22, 45, 67, 76, 95

[28] Letmanyi, H. (1985). Guide on workload forecasting, special public. In *Computer Science and Technology, National Bureau of Standards*. 69

[29] Litoiu, M. (2007). A performance analysis method for autonomic computing systems. *ACM Trans. Auton. Adapt. Syst.* 37, 38, 40

[30] Little, J. ((May - Jun., 1961)). A proof for the queuing formula: L= $\lambda$ w. *Operations Research*, **9**(3), 383–387. 21, 32

[31] Liu, Z., Squillante, M., and Wolf, J. (2001). On maximizing service-level-agreement profits. pages 213–223. 36

[32] MacKie-Mason, J. and Varian, H. (1995). Pricing congestible network resources. *IEEE Journal on Selected Area in Communications*, **13**(7), 1141–1149. 40

[33] Mahanti, A., Williamson, C., and Wu, L. (2009). Workload characterization of a large systems conference web server. In *Proceedings of the 2009 Seventh Annual Communication Networks and Services Research Conference*, pages 55–64, Washington, DC, USA. IEEE Computer Society. 63

[34] Martinich, J. (1996). *Production and Operations Management : An Applied Modern Approach*. John Wiley and Sons. 62

[35] Marzolla, M. and Mirandola, R. (2007). Performance prediction of web service workflows. *The third International Conference on the Quality of Software-Architectures (QoAS)*, pages 127–144. 22, 24

[36] Mathew, G. and Xiaojiang, D. (2010). Securing multi-tiered web applications. *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 505 –509. 8

[37] Menascé, D. (2001). Using performance models to dynamically control e-business performance. In *Proc. 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, pages 11–14. 40

[38] Menascé, D. (2003). Workload characterization. In *IEEE Internet Computing*, pages 89–92, Piscataway, NJ, USA. IEEE Educational Activities Department, v.7 n.5, 89-92. 63

[39] Menascé, D. and Almeida, V. (May 7, 2000). *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, Upper Saddle River, NJ. 31

[40] Menascé, D. and Almeida, V. (September 21, 2001). *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall, Upper Saddle River, NJ. 21, 32, 44, 45, 47, 63, 65, 66, 67, 68, 69

[41] Menasce, D. A., Almeida, V. A., Fonseca, R., and Mendes, M. A. (2000). Business-oriented resource management policies for e-commerce servers. *Performance Evaluation*, **42**(2-3), 223–239. 25, 36, 119

[42] Nau, R. F. (Accessed on 2012). Class 9: Introduction to autoregressive integrated moving average (arima) models. *http://www.duke.edu/ rnau/411arim.htm*. 71

[43] Reiser, M. and Lavenberg, S. (1980). Mean-value analysis of closed multichain queuing networks. *Journal of the Association for Computing Machinary*, **27**(2), 313–322. 21, 32

[44] Rolia, J., Zhu, X., Arlitt, M., and Andrzejak, A. (2002). Statistical service assurances for applications in utility grid environments. *Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, pages 247–256. 37

[45] Rolia, J., Zhu, X., Arlitt, M., and Andrzejak, A. (2004). Statistical service assurances for applications in utility grid environments. *Perform. Eval.*, **58**(2+3), 319–339. 21, 32

[46] Stallings, W. (2000). Queuing analysis. *www.box.com/shared/static/lu626umiib.pdf*. xvii, 19, 20

[47] Sundarapandian, V. (December 1, 2009). *Probability, Statistics and Queueing Theory*. PHI Learning. 47

[48] Surveys, G. C. W. U. (Accessed on 2012). Gvu's www surveying team graphics visualization and usability center college of computing georgia institute of technology atlanta ga 30332-0280. In *http://www.gvu.gatech.edu/user_surveys*. 62

[49] Tantawi, A. and Towsley, D. (1985). Optimal static load balancing in distributed computer systems. *J. ACM*, **32**(2), 445–465. 16

[50] Tantawi, A., Towsley, G., and Wolf, J. (1988). Optimal allocation of multiple class resources in computer systems. *SIGMETRICS Perform. Eval. Rev.*, **16**(1), 253–260. 16

[51] Urgaonkar, B., Shenoy, P., Chandra, A., and Goyal, P. (2005). Dynamic provisioning of multi-tier internet applications. In *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*, pages 217–228, Washington, DC, USA. IEEE Computer Society. 25

[52] Vercauteren, T., Aggarwal, P., Xiaodong, W., and Ta-Hsin, L. (2007). Hierarchical forecasting of web server workload using sequential monte carlo training. *Signal Processing, IEEE Transactions on*, pages 1286 –1297. 45, 68, 91, 111

[53] Wang, Q. (2004). Workload Characterization and Customer Interaction at E-commerce Web Servers. Master's Thesis, University of Saskatchewan. 3, 62

[54] Xue, J. W. J., Chester, A. P., He, L., and Jarvis, S. A. (2008). Dynamic resource allocation in enterprise systems. In *ICPADS '08: Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 203–212, Washington, DC, USA. IEEE Computer Society. xvii, 12, 13, 14, 21, 22, 25, 26, 31, 32, 35, 37, 42, 47, 50, 72

[55] Yang, C. and Luo, M. (2000). Realizing fault resilience in web-server cluster. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, Supercomputing '00, Washington, DC, USA. IEEE Computer Society. 11

[56] Yang, C. S. and Luo, M. (1998). Design and implementation of an administration system for distributed web server. In *Proceedings of the 12th Conference on Systems Administration*, pages 131–140, Berkeley, CA, USA. USENIX Association. 10

[57] Zhou, J. and Yang, T. (2006). Selective early request termination for busy internet services. *Proceedings of the 15th international conference on World Wide Web*, pages 605–614. 34, 37