Nelson Mandela
Metropolitan
University

for tomorrow

A Comparison Framework for Server Virtualisation Systems: A Case Study

Martin Stephen van Tonder

Submitted in partial fulfilment
of the requirements for the degree of
MAGISTER SCIENTIAE
in the Faculty of Science at the
Nelson Mandela Metropolitan University

January 2006

Supervisor: Mr N.L.O. Cowley

Acknowledgments

I would like to thank my supervisor Lester Cowley for his encouragement, support and guidance throughout this research. I also wish to express my gratitude to him for proposing this project, and for the many thought-provoking discussions we had over this period.

I would also like to thank Mark Ridley for his help and guidance in setting up the NMMU Telkom Centre of Excellence (CoE) data centre.

I would like to thank the CoE for providing me the opportunity to conduct this research, and for providing the resources to make this possible.

I would also like to thank the CoE industry partners: Telkom SA, Sun Microsystems SA, Dimension Data SA and THRIP for funding the research of the CoE and thus this research project.

Table of Contents

Summary v

List of Figures vii

List of Tables x

Chapter 1: Introduction 1

1.1 Background 1

1.2 Situation of Concern 1

1.3 Purpose of Research 2

1.4 Research Methodology 3

1.5 Dissertation Outline 6

Chapter 2: An Overview of Data Centres 8

2.1 Introduction 8

2.2 Definition of a Data Centre 8

2.3 Classification of Data Centres 9

2.3.1 Types of Server Farms 9

2.3.2 Types of Data Centres 12

2.4 Benefits of Data Centres 13

2.5 Data Centre Applications 14

2.5.1 Multitier Applications 15

2.6 Servers and Storage 19

Servers 19

Storage 21

2.7 Common Data Centre Problems 25

2.8 Conclusion 28

Chapter 3: Virtualisation 29

3.1 Introduction 29

3.2 Virtualisation Defined 29

3.3 Virtual Machines and Server Partitioning 30

3.4 Approaches to Virtualisation 33

3.5 Advantages of Server Virtualisation 55

3.6 Applications of Virtualisation in Data Centres 59

3.7 Conclusion 61

Chapter 4: Current Virtualisation Systems..62
4.1 Introduction..62
4.2 Current Virtualisation Systems..63
Solaris 10 Containers..63
Sun Dynamic System Domains..72
HP Node Partitions...74
HP Virtual Partitions..76
IBM Logical Partitions...81
VMware ESX Server 2.5...91
Microsoft Virtual Server 2005...96
4.3 Conclusion..99
Chapter 5: Virtualisation Systems Comparison Framework..............................101
5.1 Introduction..101
5.2 Intended Audience...103
5.3 Applications...104
5.4 Clarifications..104
5.5 Framework Criteria Selection..105
5.6 Structure of the Framework...107
5.6.1 Structure of Evaluations..108
5.7 Ranking System..109
5.8 Framework Criteria..112
5.8.1 Compatibility (C 1)...112
5.8.2 Isolation (C 2)..118
Fault Isolation (C 2.1)...119
Resource Isolation (C 2.2)...122
Security and Namespace Isolation (C 2.3)......................................131
5.8.3 Manageability (C 3)...137
5.8.4 Flexibility (C 4)..140
5.8.5 Granularity (C 5)..144
5.8.6 Scalability (C 6)..149
5.8.7 Performance (C 7) 159
Microbenchmarks 161

Application Benchmarks....161
Framework Performance Criteria....161
The Exclusion of Cost as a Framework Criterion....166
5.9 Interpreting Results and Applying the Framework....166
5.9.1 Virtualisation Requirements Filters....167
5.10 Generality and Extensibility of the Framework....172
5.11 Conclusion....173
Chapter 6: Evaluation of Existing Systems Using the Framework.... 175
6.1 Introduction....175
6.2 Evaluations....176
6.3 Strengths and Weaknesses Analysis....244
6.4 Evaluation Feasibility....250
6.5 Conclusion....251
Chapter 7: System Selection and Evaluation Case Study....254
7.1 Introduction....254
7.2 System Selection Case Study....255
7.3 System Evaluation Case Study....260
7.3.1 Test Environment....260
7.3.2 Evaluation Criteria....261
Manageability (C 3)....261
Granularity (C 5)....263
Scalability – Scaling to More Partitions (C 6.2)....263
Performance (C 7)....267
7.4 Conclusion....270
Chapter 8: Conclusions....272
8.1 Introduction....272
8.2 Evaluation of Outcomes....272
Objectives....272
Framework Design Principles....273
8.3 Significance of the Research....275
8.4 Future Research....276
References.... 278

Summary

Recent years have seen a revival of interest in virtualisation research. Although this term has been used to refer to various systems, the focus of this research is on systems which partition a single physical server into multiple virtual servers.

It is difficult for researchers and practitioners to get a clear picture of the state of the art in server virtualisation. This is due in part to the large number of systems available. Another reason is that information about virtualisation systems lacks structure, and is dispersed among multiple sources.

Practitioners, such as data centre managers and systems administrators, may be familiar with virtualisation systems from a specific vendor, but generally lack a broader view of the field. This makes it difficult to make informed decisions when selecting these systems.

Researchers and vendors who are developing virtualisation systems also lack a standard framework for identifying the strengths and weaknesses of their systems, compared to competing systems. It is also time-consuming for researchers who are new to the field to learn about current virtualisation systems.

The purpose of this research was to develop a framework to solve these problems. The objectives of the research correspond to the applications of the framework. These include conducting comparative evaluations of server virtualisation systems, identifying strengths and weaknesses of particular virtualisation systems, specifying virtualisation system requirements to facilitate system selection, and gathering information about current virtualisation systems in a structured form. These four objectives were satisfied. The design of this framework was also guided by six framework design principles. These principles, or secondary objectives, were also met.

The framework was developed based on an extensive literature study of data centres, virtualisation and current virtualisation systems. Criteria were selected through an inductive process. The feasibility of conducting evaluations using the framework was demonstrated by means of literature-based evaluations, and a practical case study. The use of the framework to facilitate virtualisation system selection was also demonstrated by means of a case study featuring the NMMU Telkom CoE data centre.

This framework has a number of practical applications, ranging from the facilitation of decision-making to identifying areas for improvement in current virtualisation systems. The information resulting from evaluations using the framework is also a valuable resource for researchers who are new to the field. The literature study which forms the theoretical foundation of this work is particularly useful in this regard.

A future extension to this work would be to develop a decision support system based on the framework. Another possibility is to make the framework, and evaluations, available on-line as a resource for data center managers, vendors and researchers. This would also enable other researchers to provide additional feedback, enabling the framework to be further refined.

Keywords: data centre; server partitioning; server virtualisation; virtual machines; virtualisation.

List of Figures

Figure 2.1: An Intranet Server Farm....10
Figure 2.2: An Extranet Server Farm.... 11
Figure 2.3: An Internet Server Farm....12
Figure 2.4: An Enterprise Data Centre.... 13
Figure 2.5: Two-tier, Three-tier and N-tier Architectures....16
Figure 2.6: Tiers of an N-tier System.... 17
Figure 2.7: Storage Area Network (SAN) Overview.... 22
Figure 2.8: A Storage Area Network....23
Figure 2.9: Network Attached Storage Overview.... 24
Figure 2.10: Server Attached Storage Overview....25
Figure 3.1: Virtual Machine Monitor Hosting Virtual Machines....31
Figure 3.2: A Virtual Machine Taxonomy.... 33
Figure 3.3: A Taxonomy of Partitioning Techniques....34
Figure 3.4: A Native Virtual Machine Monitor....35
Figure 3.5: User-Mode Hosted Virtual Machine Monitor....36
Figure 3.6: A Dual-Mode Hosted Virtual Machine Monitor....38
Figure 3.7: Ticket Exchange....47
Figure 3.8: One Process Per Connection....49
Figure 3.9: One Thread Per Connection....49
Figure 3.10: One Process for All Connections....50
Figure 3.11: Processing Accounted Incorrectly....50
Figure 3.12: A Sun Server Physically Partitioned into Two Domains.... 53
Figure 3.13: Consolidation and Server Utilisation....56
Figure 4.1: Zone CPU Resource Allocation....64
Figure 4.2: Zones Shares Allocations....65
Figure 4.3: Solaris Management Console....68
Figure 4.4: Filesystem Subtrees for Non-global Zones....69
Figure 4.5: A Server Hosting Various Applications within Zones....71
Figure 4.6: Zone Performance Relative to Non-Zone Performance....72
Figure 4.7: A System Board.... 73
Figure 4.8: Configuration before DR Operation.... 74

Figure 4.9: Configuration after DR Operation... 74
Figure 4.10: A Hewlett Packard Server with Two Cell Boards.................................. 75
Figure 4.11: Virtual Partitions...77
Figure 4.12: vPar Software Isolation and Compatibility... 78
Figure 4.13: Assigning Resources to Virtual Partitions.. 79
Figure 4.14: Virtual Partitions...80
Figure 4.15: Hardware Management Console...82
Figure 4.16: Dedicated Processor Partitions..83
Figure 4.17 The Dispatch Wheel..84
Figure 4.18: Mixing Partition Types... 85
Figure 4.19: Virtual SCSI..86
Figure 4.20: Virtual Ethernet...87
Figure 4.21: A Capped Partition..88
Figure 4.22: An Uncapped Partition...88
Figure 4.23: The Architecture of VMware ESX Server...92
Figure 4.24: Balloon Driver...93
Figure 4.25: Memory Tax Formula.. 94
Figure 4.26: Architecture of Virtual Server 2005...97
Figure 4.27: Virtual Server Management Interface..99
Figure 5.1: The First Two Levels of the Framework Structure.................................107
Figure 5.2: Compatibility Criteria... 113
Figure 5.3: Types of Isolation...119
Figure 5.4: Fault Isolation Criteria.. 120
Figure 5.5: Resource Isolation Criteria..124
Figure 5.6: Security and Namespace Isolation Criteria.. 133
Figure 5.7: Manageability Criteria... 138
Figure 5.8: Flexibility Criteria..141
Figure 5.9 : Resource Allocation Granularity...144
Figure 5.10: Granularity Criteria.. 145
Figure 5.11: Two Types of Scalability.. 149
Figure 5.12: Criteria for Scalability – Scaling to Larger Partitions...........................150
Figure 5.13: Criteria for Scalability – Scaling to More Partitions.............................153

Figure 5.14: Apache Performance....160
Figure 6.1: Ranking Results: Compatibility Category....186
Figure 6.2: Ranking Results: Fault Isolation Category....189
Figure 6.3: Ranking Results: Resource Isolation Category....197
Figure 6.4: Ranking Results: Security and Namespace Isolation Category....202
Figure 6.5: Ranking Results: Manageability Category....208
Figure 6.6: Ranking Results: Flexibility Category....215
Figure 6.7: Ranking Results: Granularity Category....223
Figure 6.8: Ranking Results: Scalability – Scaling to Larger Partitions Category....230
Figure 6.9: Ranking Results: Scalability – Scaling to More Partitions Category....243
Figure 7.1: Time Taken to Install a Zone....262
Figure 7.2: Total Memory Usage....264
Figure 7.3: Memory Usage With 15 Zones Booted....265
Figure 7.4: Disk Space Consumed by Zones....266
Figure 7.5: Apache Throughput (Zone versus Global)....268
Figure 7.6: Sysbench OLTP Throughput....270

List of Tables

Table 5.1: Response Data Types.......... 108
Table 5.2 Definition of a Filter Production Rule.......... 170
Table 6.1: Instruction Set Architectures Supported.......... 178
Table 6.2a: Operating System Compatibility.......... 179
Table 6.2b: Operating System Compatibility.......... 180
Table 6.3a: Device Compatibility.......... 181
Table 6.3b: Device Compatibility.......... 182
Table 6.4: Compatibility with Existing Applications.......... 183
Table 6.5a: Concurrent Operating Systems Compatibility.......... 184
Table 6.5b: Concurrent Operating Systems Compatibility.......... 185
Table 6.6: Physical Isolation Between Partitions.......... 187
Table 6.7: Separate Operating System Kernel for Each Partition.......... 188
Table 6.8: CPU Resource Isolation.......... 191
Table 6.9a: Memory Resource Isolation.......... 192
Table 6.9b: Memory Resource Isolation.......... 193
Table 6.10: Disk Bandwidth Isolation.......... 194
Table 6.11: Disk Space Usage Limits.......... 195
Table 6.12: Network Bandwidth Isolation.......... 196
Table 6.13: Filesystem Isolation.......... 199
Table 6.14: Isolation of Package Databases.......... 199
Table 6.15: Isolation of Network Port Bindings.......... 200
Table 6.16: Inter-partition Communication Isolation.......... 200
Table 6.17: Isolation of Operating System Tuning Parameters.......... 201
Table 6.18: Separate Set of Users for Each Partition.......... 201
Table 6.19a: GUI Tools for Partition Management.......... 204
Table 6.19b: GUI Tools for Partition Management.......... 205
Table 6.20: Scripting Support.......... 206
Table 6.21: Live Migration of Partitions Between Separate Servers.......... 207
Table 6.22: CPU Reallocation Without Reboot.......... 211
Table 6.23: Memory Reallocation Without Reboot.......... 212
Table 6.24a: Tools to Automate Resource Allocation.......... 213

Table 6.24b: Tools to Automate Resource Allocation.. 214
Table 6.25a: Memory Resource (Re)Allocation Quantity...217
Table 6.25b: Memory Resource (Re)Allocation Quantity...218
Table 6.26a: CPU Resource (Re)Allocation Quantity...219
Table 6.26b: CPU Resource (Re)Allocation Quantity.. 220
Table 6.27: CPU Reallocation Time Granularity.. 221
Table 6.28: Memory Reallocation Time Granularity.. 222
Table 6.29a: Maximum CPUs Per Partition..225
Table 6.29b: Maximum CPUs Per Partition..226
Table 6.29c: Maximum CPUs Per Partition..227
Table 6.30a: Maximum Memory Per Partition..228
Table 6.30b: Maximum Memory Per Partition... 229
Table 6.31: Maximum Partitions Per CPU..233
Table 6.32: Sharing of Network Adapters...234
Table 6.33: Sharing of Disk Resources... 234
Table 6.34a: Memory Consumption..235
Table 6.34b: Memory Consumption..236
Table 6.34c: Memory Consumption..237
Table 6.35a: Disk Space Consumed by a Partition... 238
Table 6.35b: Disk Space Consumed by a Partition... 239
Table 6.36: Pooling of Memory... 240
Table 6.37a: Fixed Limits on Number of Partitions Per System...............................241
Table 6.37b: Fixed Limits on Number of Partitions Per System...............................242
Table 7.1: Test Server Configuration... 260
Table 7.2: The Configuration of the V240 Test Client... 260
Table 7.3 Time Taken to Install a Partition...261
Table 7.4: Memory Usage with Fifteen Zones Booted..265
Table 7.5 Apache Throughput for Virtualised and Non-Virtualised Cases................267
Table 7.6: Sysbench OLTP Transactions Per Second (tps).......................................269

Chapter 1: Introduction

1.1 Background

In 1966 a specially modified IBM System/360 Model 40 became the first operational virtual machine system [Cre1981]. With this system, a single physical machine was shared by multiple virtual machines. From the perspective of a user, or application, each of these virtual machines appeared identical to an independent System/360.

Nearly four decades later, the design of this system and its successors is still influencing the design of current systems. All of these systems are based on the same principle, the principle of virtualisation. Informally, virtualisation involves transforming a system to provide the illusion of a different system, or multiple systems [SN2005]. This term has been used to describe various concepts, including grid computing, pooling of storage resources, and virtual machines, among others. For the purposes of this document, the term virtualisation will refer to the partitioning of a single physical server into multiple logical servers.

Recently there has been an increase in virtualisation research by both academic researchers and vendors. This has been driven by the need for server consolidation. Xen [BDF+2003] and Denali [WSG2002a] are just two examples of virtualisation systems originating in academia. Vendors such as IBM [IBM2004b], Sun [SM2004d], Hewlett Packard [HP2004e], VMware [VMW2004a] and Microsoft [MS2004b] (among others) are also actively working in this area. Virtualisation systems from these vendors are intended for use in commercial data centres. This application of virtualisation will be the focus of this research.

1.2 Situation of Concern

Virtualisation systems are increasingly being used in data centres. Virtually all of the major vendors provide their own virtualisation systems. In addition, academic researchers, open source developers and startup companies have also been actively developing new systems. This has led to a rapid increase in the number of virtualisation systems available. Virtualisation functionality which was once restricted to high-end computing equipment, such as mainframes, is now available for low cost servers, and even desktop systems. This has resulted in users with limited knowledge of virtualisation being exposed to these systems. Currently, users include practitioners, such as data centre managers and systems administrators, and technical desktop users. Many of these users have a very limited understanding of virtualisation. Despite this, they are tasked with evaluating the virtualisation systems available, and making a system selection decisions.

Developers of these systems, including researchers and vendors, are finding it increasingly difficult to keep track of the capabilities provided by competing systems. As a result, it is difficult to determine the state of the art in server virtualisation. This is also true for researchers who are not involved in the development of these systems, especially those who are new to the field. Numerous system manuals, research papers, and other sources need to be consulted. Locating these documents, and understanding the information contained therein, is a non-trivial task.

To the best of the author's knowledge there is currently no independent framework for evaluating and comparing server virtualisation systems. The lack of a standard framework makes it difficult for practitioners, researchers and vendors to determine the relative strengths and weaknesses of these systems.

1.3 Purpose of Research

A solution to these problems would be to develop an independent framework for evaluating server virtualisation systems, in order to conduct comparisons. The development of such a framework is the aim of this research. This framework should consist of criteria for evaluating these types of systems. A framework would be of practical benefit to both researchers and practitioners.

Researchers who are new to the field would be able to examine data from evaluations conducted using the framework. This would greatly simplify the task of learning about the functionality provided by current systems.

Researchers and vendors involved in the development of virtualisation systems could use the framework to evaluate their systems. This would enable them to easily identify the strengths and weaknesses of their systems relative to competing systems.

Identifying strengths and weaknesses would also be useful for practitioners. Decision-makers, such as data centre managers, need to identify strengths and weaknesses in order to select systems which are best suited to their requirements. To facilitate this application the framework should include a mechanism for defining requirements in terms of the framework criteria.

The major virtualisation systems are designed for use in data centres, and are of considerable economic importance. For this reason, the focus of this framework will be on virtualisation systems designed for this environment.

The following list summarises the objectives of this research:

- Developing a framework for evaluating server virtualisation systems in order to conduct comparisons
- Facilitating identification of system strengths and weaknesses using the framework
- Providing a means to define requirements based on the framework criteria in order to facilitate system selection
- Gathering information about current virtualisation systems, in the form of evaluations using the framework, in order to facilitate learning by novices

1.4 Research Methodology

The first step in constructing the framework was to conduct a preliminary literature study in the planning phase of the research. It became clear that this research would require an understanding of data centres, virtualisation and current virtualisation systems. In order to achieve this, an extensive literature study will be conducted. This literature study forms the foundation of the framework.

Based on the preliminary literature study, the following topics were identified for further investigation to inform the selection of framework criteria.

- Types of data centres and data centre applications.
- Hardware and operating system software commonly found in data centres.
- Common data centre problems.
- Virtualisation theory and principles.
- Approaches to implementing virtualisation systems, and the trade-offs involved.
- Resource management and virtualisation.
- Advantages of virtualisation.
- Applications of virtualisation in data centres.
- Functionality provided by current virtualisation systems.

The framework consists of a number of categories. These categories were identified by a process of induction. The body of information gathered about virtualisation systems in the literature study was considered. This was to identify how it could be categorically structured in a logical manner. Each of the categories consists of lower level criteria. These criteria represent characteristics which are identified as the most significant factors affecting the higher level characteristic.

Framework Design Principles

The selection of the framework criteria was also influenced by the six design principles which are discussed next.

- **Objective:**
  As noted previously, there is currently no independent framework for comparing server virtualisation systems. Comparisons provided by vendors cannot realistically be considered objective. The criteria selected for the framework should not be selected to favour any particular system.

- **Thorough**
  In order to compare systems effectively, there need to be enough criteria used in the comparison. A good framework should include criteria which highlight both the strengths and weaknesses of all systems, without being biased.
- **Balance Between Thoroughness and Conciseness**
  Making the framework too thorough, or detailed, could make it difficult to interpret, and therefore restrict its usefulness. This would also complicate the process of evaluating a system using the framework. Considering too few criteria could render the framework ineffective for conducting comparisons. The framework needs to strike a balance between these two considerations.
- **Generic**
  The framework should be generic, enabling a wide range of virtualisation systems to be evaluated.
- **Extensible**
  The structure of the framework should be extensible, enabling additional criteria to be added in future.
- **Relevance to Data Centres**
  As stated previously, the focus of the framework is on virtualisation in the data centre. The criteria which are selected need to be selected with this in mind.

A means to specify requirements in terms of the framework criteria is also developed, based on expert systems literature.

In order to create a body of knowledge based on the framework, a number of current virtualisation systems are evaluated. For practical reasons a literature study is the most appropriate method for doing this. This approach is used to evaluate ten current virtualisation systems. Three additional systems not considered during the initial literature study phase of the research are also evaluated to demonstrate the generality of the framework.

These evaluations are also used to demonstrate the practicality of conducting evaluations using the framework criteria. The information gathered is used to test the usefulness of the framework in identifying strengths and weaknesses of systems.

Evaluating systems based on literature has limitations. Practical aspects of the framework, such as performance measurement, cannot be evaluated by studying system manuals and research papers. In order to demonstrate this application of the framework a case study is conducted. Solaris Containers is evaluated using the facilities of the NMMU Centre of Excellence (CoE) data centre as a case study. This case study, together with the literature-based evaluations, provides an example for other researchers to follow when evaluating other systems using the framework.

One of the applications of the framework is to provide decision-makers with a means to define requirements in terms of the framework criteria. This is to facilitate system selection based on these requirements. The NMMU Telkom CoE data centre is used as a case study to demonstrate this application of the framework.

1.5 Dissertation Outline

Chapter 2 provides an overview of data centres. This study examines the types of data centres, servers, data centre applications and common data centre problems. This chapter is important because the focus of this research is on virtualisation systems designed for use in data centres.

Chapter 3 explores the concept of virtualisation. This chapter is based on research papers and literature, some of which date back to the 1970s. Various approaches to implementing virtualisation systems, and advantages and applications of virtualisation in the data centre, are identified in this chapter. These applications are used to motivate the selection of framework criteria in later chapters.

Seven current virtualisation systems are reviewed in Chapter 4. In this chapter the features and capabilities provided by current virtualisation systems are explored. This information is needed to select framework criteria which are relevant. These reviews are based primarily on system manuals and research papers. This study includes some of the most widely used virtualisation systems available. This information was particularly useful in the selection of some of the lower level criteria of the framework in Chapter 5.

Chapter 5 presents the framework which is the subject of this research. The goals, intended audience and applications of the framework are also presented in this chapter. A ranking system intended to aid in the interpretation of evaluations is also presented. The concept of a virtualisation requirements filter is also defined in this chapter. This type of filter is for specifying requirements based on the framework criteria.

In Chapter 6 ten major virtualisation systems are evaluated using the framework. These evaluations provide an example for researchers who wish to evaluate other systems using the framework. Another important reason for these evaluations is to populate a body of knowledge structured around the framework. The ten virtualisation systems evaluated include most of the major virtualisation systems which are commercially available today. For practical reasons these evaluations are based on literature. The strengths and weaknesses of the ten systems are also identified.

In Chapter 7 a virtualisation requirements filter is used to define requirements in order to select a suitable virtualisation system for the NMMU Telkom CoE data centre. This data centre is also used to conduct a complete evaluation of the selected system (Solaris Containers), including practical aspects. This evaluation complements the literature-based evaluations in Chapter 6 by providing an example of a complete evaluation.

Chapter 8 concludes this dissertation and reflects on the significance of this work. The extent to which the objectives of the project were met is also discussed. A number of possible future research directions are explored.

Chapter 2: An Overview of Data Centres

2.1 Introduction

This chapter is the first of three literature study chapters used to inform the design of the framework. The purpose of this chapter is to provide an overview of data centres.

Data centres are of interest to this study because server virtualisation systems are designed to partition servers, such as those found in data centres. Virtualisation systems are being promoted by server vendors such as HP [HP2004e], IBM [IBM2004b] and Sun [SM2004d] as well as others as a means to solve many of the problems experienced in data centres today. In order to understand the usefulness of these systems it is necessary to understand the environment in which they are deployed.

This chapter includes the following sections:

- Definition of a data centre
- Classification of data centres
- Benefits of data centres
- Data centre applications
- Servers and storage
- Common data centre problems

2.2 Definition of a Data Centre

There are numerous definitions of a data centre. A few examples are listed here:

"A data center is an application development implementation work product consisting of a facility housing one or more production environments (e.g. server computers, network connectivity devices, databases, applications) that is used by the operations organization to perform data processing for end user organizations"
-Firesmith [Fir2002].

"Data Centers are physical locations that house critical computing resources. Data Centers exist to support business critical applications and their attendant computing resources such as mainframes, servers, and server farms"
- Cisco [CS2002a]

"Data centers are large computing facilities with centralized resources and management infrastructure"
- Andrzejak *et al*. [AAR2002]

Based on these definitions the following definition is synthesised:

A Data Centre is a dedicated, centralized, secure computing facility housing infrastructure (both hardware and software) used to host data and applications for organisations.

This study will focus on the compute equipment and applications hosted by data centres rather than the physical environment which houses this equipment and applications.

2.3 Classification of Data Centres

Data centres are classified by Cisco [CS2002a] based on the types of server farms they support. There are three types; intranet, extranet and Internet, server farms. A data centre may support more than one type of server farm.

2.3.1 Types of Server Farms

Intranet Server Farms

Intranet server farms host applications, often web-based, for users within an organisation. Intranets are often used to make internal documents and memos available to employees [Sch2003]. Figure 2.1 depicts an intranet server farm. The symbols used in this and subsequent diagrams are standard network diagram symbols. The SP1 and SP2 clouds in this diagram represent two service providers. Users can access the intranet from within the organisation (campus) or off campus. Off campus access is achieved using VPN (Virtual Private Network) technology to access the intranet over the public Internet [CS2002a].

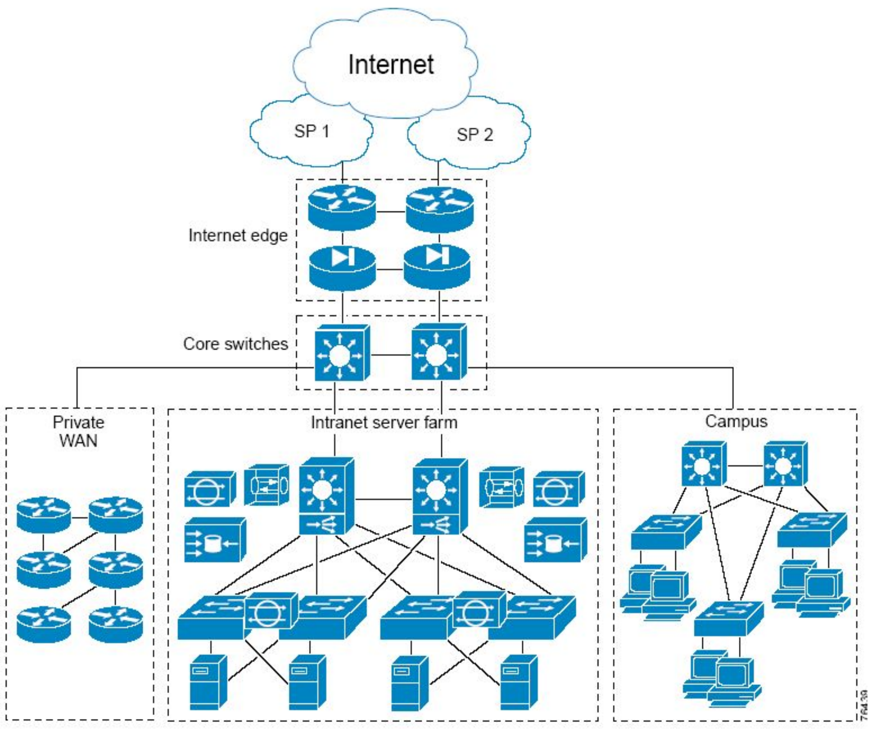


Figure 2.1: An Intranet Server Farm [CS2002a]

Extranet Server Farms

Extranet server farms can be viewed as an extension of the intranet server farm. This type of farm hosts applications which can be accessed by employees and a selected set of partner organisations such as suppliers. Figure 2.2 depicts an extranet server farm.

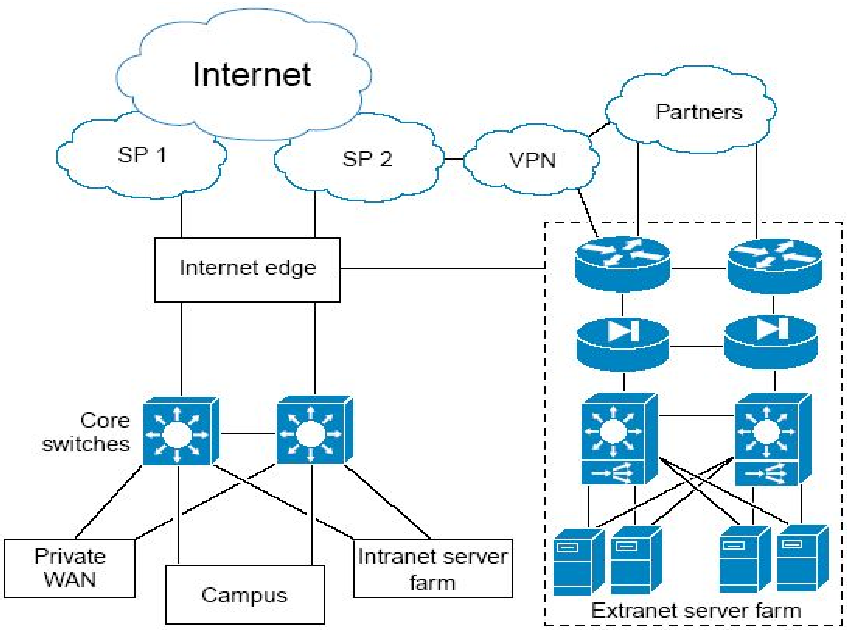


Figure 2.2: An Extranet Server Farm [CS2002a]

Internet Server Farms

Internet server farms host web applications accessible to users on the Internet [CS2002a]. Examples include transaction processing applications for business-to-business or business-to-consumer transactions [Sch2003]. Such sites require 24 hours a day, seven days a week availability [Sch2003]. Due to the large number of potential users such farms need to be scalable [CS2002a]. Scalability refers to the ability to deal with higher volumes of traffic, due to user growth, by adding additional hardware.

Figure 2.3 depicts a dedicated Internet server farm.

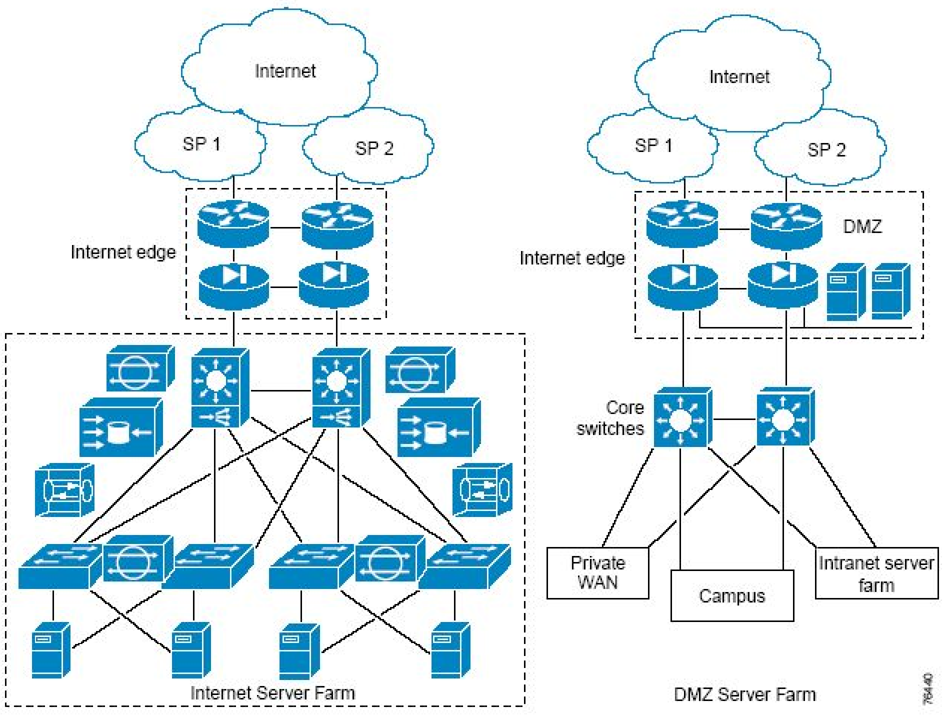


Figure 2.3: An Internet Server Farm [CS2002a]

2.3.2 Types of Data Centres

Data centres can be classified according to the type of server farms they host. Based on this classification there are two basic types of data centres:

Internet Data Centres (IDCs)

A data centre dedicated to hosting Internet server farms is known as an Internet Data Centre (IDC). As noted by Arregoces and Portolani [AP2004] IDCs are owned by enterprises and service providers. Service provider IDCs typically have higher scalability requirements due to the larger number of users served and services provided, according to Schneider [Sch2003].

Enterprise (Corporate) Data Centres

Enterprise data centres usually need to support a wide range of applications, often a mixture of Internet, intranet and extranet applications depending on the needs of the business [AP2004]. This results in many enterprise data centres supporting a mixture of Internet, intranet and extranet server farms. A typical enterprise data centre is depicted in Figure 2.4. The wide variety of hardware and software systems commonly found in data centres is clearly depicted in this figure. Data centre hardware and software are discussed in more detail in subsequent sections.

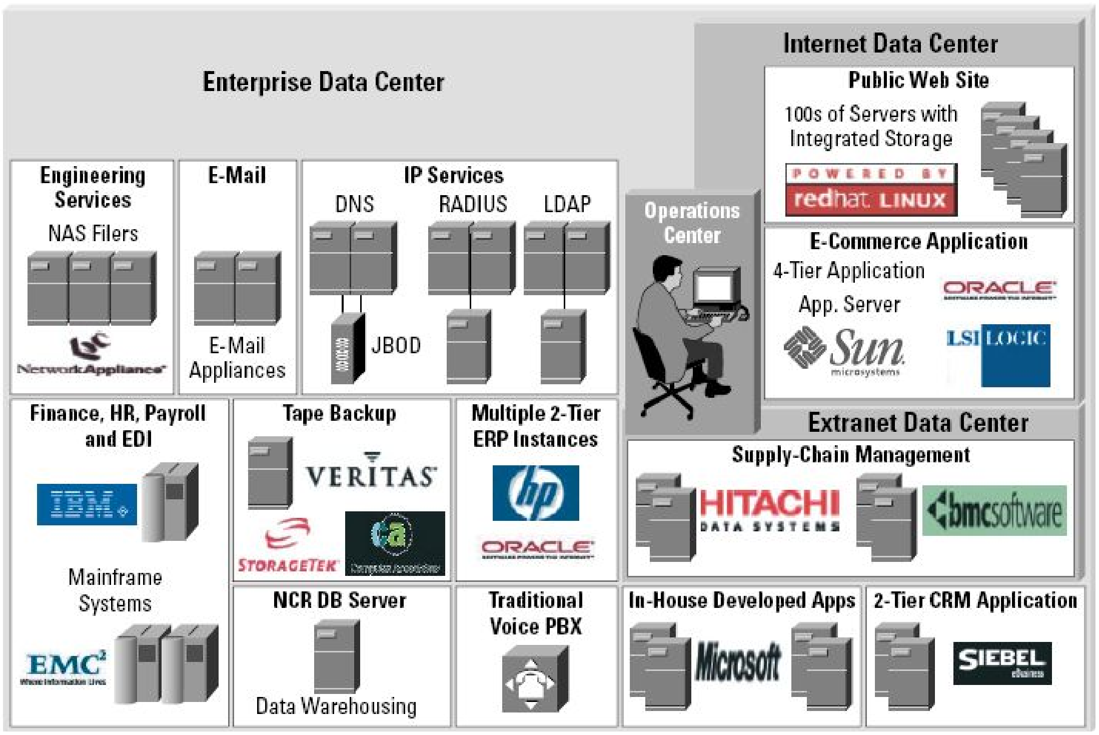


Figure 2.4: An Enterprise Data Centre [AP2004]

2.4 Benefits of Data Centres

Data centres were created to centralize computing resources and their management, according to Mehra [PM2002]. Typical benefits of data centres include:

- Improved standardisation [Fir2002]
- End users require less powerful desktop systems [Fir2002]

- Economies of scale reduce overall costs, specifically maintenance costs [Fir2002], [Kot2001]
- Improvements in scalability, security and availability [Fir2002], [Kot2001]

2.5 Data Centre Applications

According to Cisco [CS2002a] typical data centre applications include enterprise applications such as:

- Supply chain management (SCM)
  e.g. I2 Technologies and Manugistics - (Schneider [Sch2003]).
- Enterprise resource planning (ERP)
  e.g. SAP, PeopleSoft, Baan, Oracle and J.D. Edwards - (Schneider [Sch2003]).
- Customer relationship management (CRM) e.g. Siebel
- Data warehousing
- E-commerce
- Portals [AP2004]
- Business to Business (B2B) applications [Sim2004]
- Computer aided design / computer aided manufacturing (CAD/CAM [Sim2004])
- Sales force automation (SFA) (Cisco [CS2003a]).

and communications applications such as:

- IP telephony
- Voice-mail
- E-mail
- Legacy telephone
- Videoconferencing
- Instant messaging
- Media on demand (MoD) (Cisco [CS2003a]).

2.5.1 Multitier Applications

The majority of current enterprise applications such as those listed in this section are multitiered by design [AP2004]. This architecture will now be discussed in more detail due to its importance.

Multitier applications are separated into logically distinct tiers. Each tier is classified by the functional role it performs [KC2003]. Applications evolved from monolithic mainframe applications where dumb terminals were connected to mainframes, which performed all of the processing, to two-tier applications. Using a two-tier design the client tier containing application logic connects to a centralised application/database system. Due to shortcomings of the two-tier approach, the three tier and later N-tier approaches were adopted [SM2000]. One of the shortcomings of two-tier systems was a lack of scalability. This is due to each client accessing the database directly. Three tiered systems which offload much of the processing duties onto one or more application servers in the application tier offer much better scalability. Thick clients, which still perform some processing, connect to an application server which accesses the data in the database using standard interfaces [CS2003a].

Web-based applications, in which a web server connects to an application server, which in turn connects to a database server, are classic N-tier applications. The web browser of the client acts as a thin client providing another tier. One of the major advantages of a web-based N-tier application is that users do not need to install any special software [CS2003a] as only a standard web browser is required. Any system with more than three tiers is referred to as an N-tier system. The N-tier architecture is discussed in more detail in the next section.

The two-tier, three-tier and N-tier architectures are depicted in Figure 2.5.

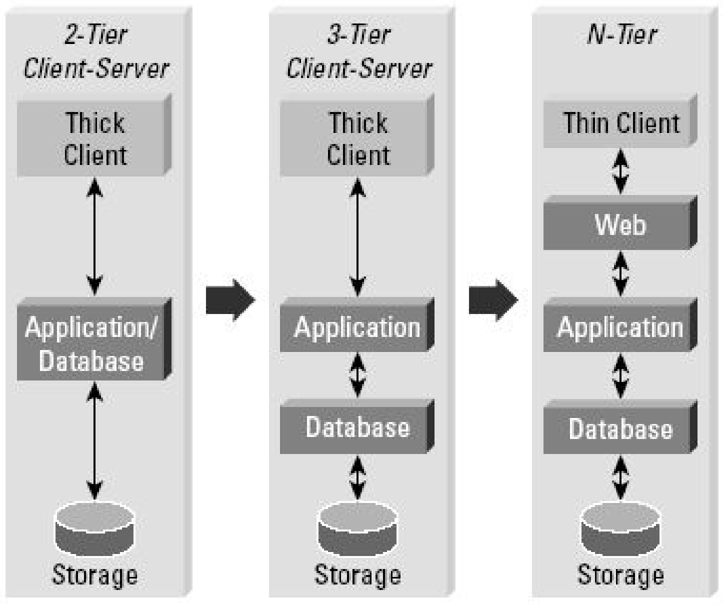


Figure 2.5: Two-tier, Three-tier and N-tier Architectures [CS2003a]

The N-tier Architecture

N-tier systems offer a number of advantages such as availability, manageability and improved resource utilisation [SM2000]. Scalability is also improved [SM2000] as the processing load can now be spread across more servers. It is possible to host all of the tiers on the same server, or to map each tier onto one or more separate servers. The separation of systems into logical tiers also enables staff to maintain each tier separately.

Figure 2.6 depicts the logical tiers of an N-tier system. Each of these tiers will now be discussed individually.

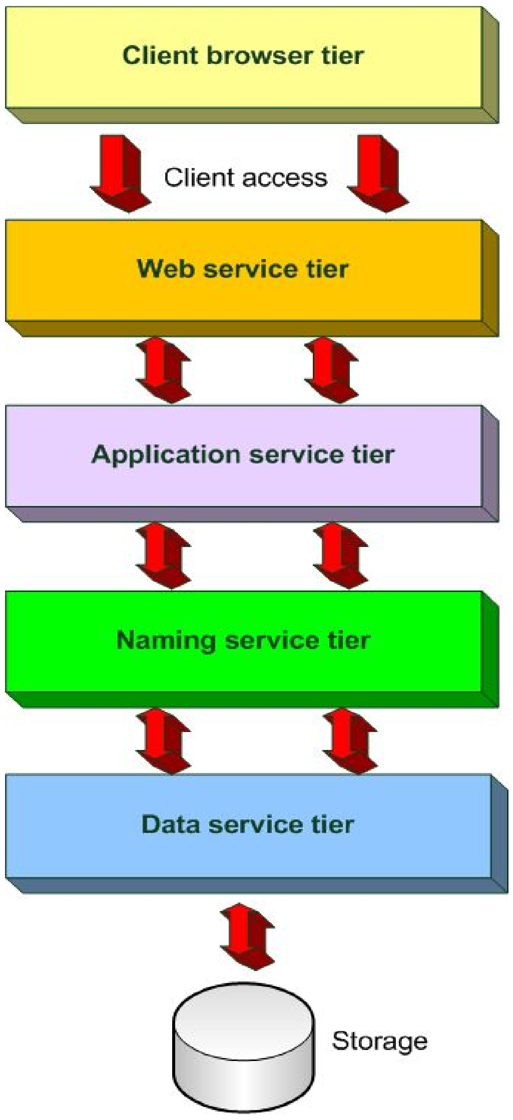


Figure 2.6: Tiers of an N-tier System (based on figure in [KC2003]*)*

Web Service Tier (Presentation Tier)

The web service tier typically serves as the presentation tier of the application, presenting users with a graphical user interface (GUI) [SM2000]. This is accessed by end users from the client browser tier. This is depicted in Figure 2.6. This tier, which is often simply referred to as the web tier, receives incoming HTTP requests from clients [KC2003]. Load balanced servers running web server software, such as Apache [ASF2005a] or Microsoft Internet Information Services (IIS), handle these requests. A HTTP response containing the requested data is sent back to the client.

Load balancing involves spreading the traffic load generated by clients over a number of servers. Each server handles a subset of the incoming requests. Dynamic web applications implemented using technologies such as JSP/Servlets, ASP, PHP or CGI can generate customised content based on user requests.

Application Service Tier

Application server software hosted by one powerful server or several smaller servers is tasked with performing the application logic [KC2003]. The application server performs processing in response to requests from the presentation tier [AP2004]. This tier connects to the data service tier to read and write data to persistent storage. Standards such as ODBC or JDBC are used to communicate with a database. Examples of application servers include BEA Weblogic Server [BEA2005], IBM WebSphere Application Server [IBM2005i] and Oracle Application Server [OC2005b]. The application tier can also connect to other systems such as SAP or legacy software [KC2003]. Often the application service tier is just referred to as the application tier.

Naming Service Tier

The naming service tier is regarded as a separate tier in an N-Tier design by Kakadia and Croucher [KC2003]. This is not always classified as a separate tier [AP2004]. Naming service servers store various types of information such as host addressing data and other settings.

Data Service Tier

This tier stores and manages application data [KC2003]. This data can be accessed by the application tier [AP2004]. Security is very important for the data tier to prevent any unauthorized access to data [KC2003]. Relational database software is generally used to manage this data. Examples of database server software include Oracle Database Server [OC2005a], IBM DB2 [IBM2005g] and Microsoft SQL Server [MS2005b] among others.

2.6 Servers and Storage

In addition to software components such as enterprise applications and databases, data centres also consist of physical elements such as servers, storage and networking equipment. Although networking equipment plays an important role in any data centre, a detailed discussion of the various network devices and topologies is beyond the scope of this study. The focus is on servers and their role in the data centre. The different types of storage are discussed because this equipment can be seen as a logical extension of the servers, providing access to data locally or over a network.

Servers

Examining applications from a logical tiered view is one perspective. Another perspective is a server-centric perspective. The applications and services provided by a data centre are dependent on the underlying hardware to perform processing tasks. Servers have different instruction set architectures (ISAs). Instruction sets commonly used by servers include x86 (IA32), x86_64 [AMD2001], POWER [IBM2005d], PA-RISC [HP2005b], SPARC [SI2005] and IA64 (Itanium) among others. The role a server fulfils depends on the type of server software installed on it. Each physical server can perform more than one function. There are many different types of servers. An application-centric list of common server types is presented by Arregoces and Portolani [AP2004]:

Web Servers

The role of web servers is to provide presentation layer functionality by responding to HTTP requests.

Application Servers

Application servers perform the bulk of the logic and processing tasks of an application. This corresponds to the application tier discussed previously.

Database Servers

Database servers runs database server software such as Oracle or IBM DB2 in order to provide persistent storage of data.

E-mail Servers

Email is an example of a collaboration application. Email servers store and forward email messages which are accessed by end users on the client using user agent software. Examples of e-mail server software includes Microsoft Exchange and Sendmail.

File Servers

File servers store files which can be accessed by users and other servers over a network. The Network File System (NFS) and Common Internet File System (CIFS) protocols are used. NFS is used by Unix systems and CIFS by Windows systems.

Directory Servers

Information about users, servers and other components is stored by directory servers. The Lightweight Directory Access Protocol (LDAP) is an example of a protocol used to access this information. Oracle Internet Directory and Microsoft Active Directory are examples of directory servers.

DNS Servers

Domain Name Service (DNS) servers translate domain names into IP addresses. Cisco Network Registrar and Microsoft DNS server are examples of DNS servers.

DHCP Servers

Clients such as workstations on a network are provided with IP addresses by Dynamic Host Configuration Protocol (DHCP) servers. IBM DHCP Server and Cisco Network Registrar are examples of DHCP servers.

RADIUS and AAA Servers

Remote Access Dial-In User Service (RADIUS) and Authentication, Authorization and Accounting (AAA) servers are used to authenticate users. These users include dial-up users, virtual private network (VPN) facilities users and desktop users among others. Microsoft Windows 2000 Internet Authentication Server (IAS) and Cisco Secure Access Control Server (CSACS) are examples of authentication servers.

Certificate Authority (CA) servers

Public keys are distributed by CA servers for applications such as E-commerce or virtual private networks (VPNs), which require these keys for secure socket layer (SSL) or IP Security (IPsec) operations. Examples include Microsoft Windows CA Server and Entrust Server.

Streaming Servers

Streaming servers provide streamed video content to users. Apple QuickTime Streaming Server and Progressive Real Server are examples of streaming servers.

TN3270 Servers

TN3270 servers are used to provide Telnet access to IBM mainframe applications. IBM mainframes make use of the Systems Network Architecture (SNA) protocol. In order for TCP/IP clients to access such mainframes, TN3270 servers can communicate using both protocols.

Storage

The storage and management of data is essential to any organization. Data is acquired over time and is often irreplaceable and unique to an organisation [IBM2003c]. The need for capacity to store this data is growing exponentially [Int2004a]. Storage options available to data centres include:

- Storage Area Networks (SAN)
- Network Attached Storage (NAS)
- Server Attached Storage (SAS)

Storage Area Networks (SAN)

The need to manage increasing volumes of data and numerous storage devices has led to the adoption of Storage Area Networks (SAN) [IBM2003c]. The Storage Network Industry Association (SNIA) defines SAN as *“a network whose primary purpose is the transfer of data between computer systems and storage elements”* [IBM2003c].

Storage elements are dedicated storage devices such as disk arrays or tape backup systems. SANs enable organisations to centralise the storage of data [IBM2003c]. This enables storage devices to be decoupled from servers [Int2004a].

A number of advantages of SANs are listed by IBM [IBM2003c]:

- Improved performance of applications
- Consolidation of storage
- Improved availability of applications due to redundant paths to data
- Centralised data easier to manage.

A number of other advantages and characteristics of a SAN are listed in Figure 2.7.

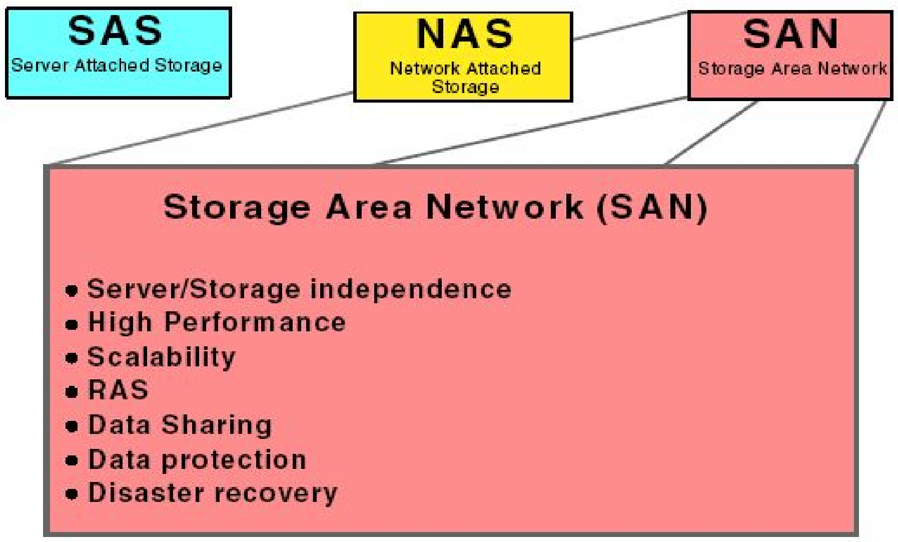


Figure 2.7: Storage Area Network (SAN) Overview [IBM2003c]

Fibre Channel technology is often used to connect servers to storage devices [AP2004]. Fibre Channel makes use of Fibre Channel Protocol to issue Small Computer Systems Interface (SCSI) commands over fibre optics (copper also an option) [Int2004a]. Due to the cost of Fibre Channel switches standards such as iSCSI and Fibre Channel over IP (FCIP) are emerging to enable IP infrastructure to be used instead.

Figure 2.8 depicts a storage area network connected using Fibre Channel switches. The servers are connected to the SAN which provides tape backup and a pool of storage resources.

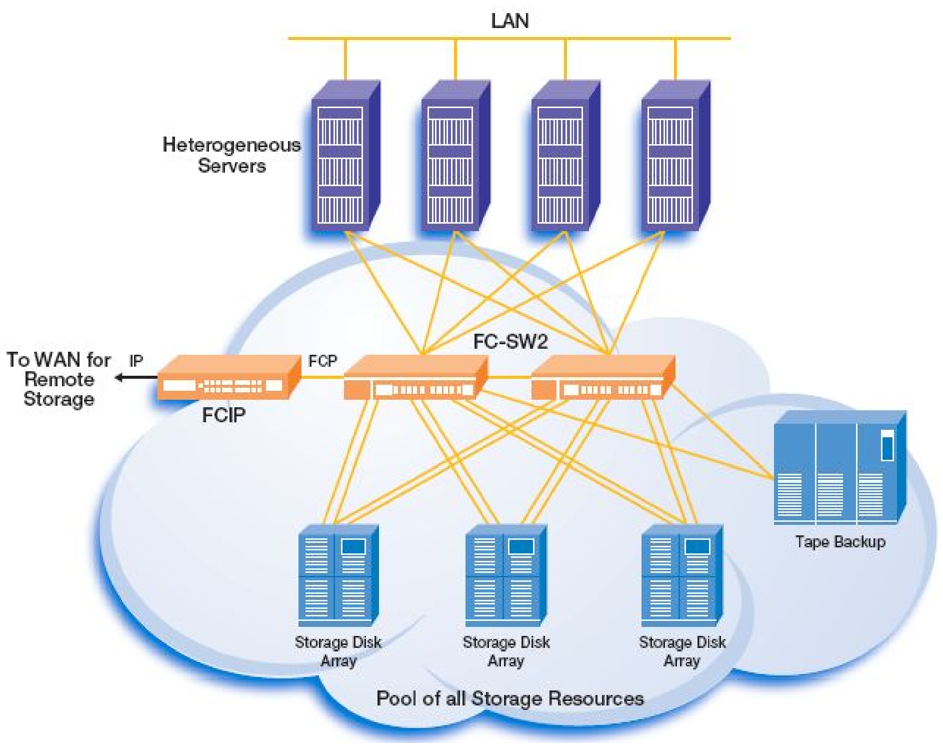


Figure 2.8: A Storage Area Network [Int2004a]

Network Attached Storage (NAS)

Another approach to shared storage is Network Attached Storage (NAS). NAS makes use of the Network File System (NFS) for Unix and Common Internet File System (CIFS) protocols for Windows to share files with other servers. A NAS server is a file server attached to a LAN which enables other servers to access files over a network [IBM2003c].

Figure 2.9 lists some of the attributes of NAS systems.

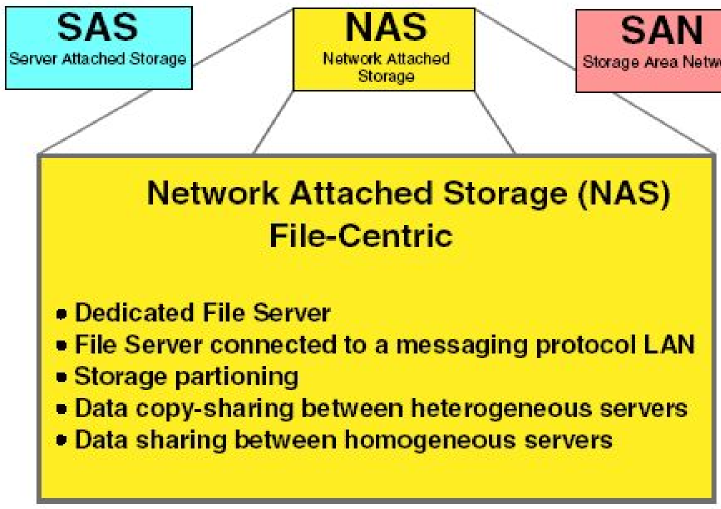
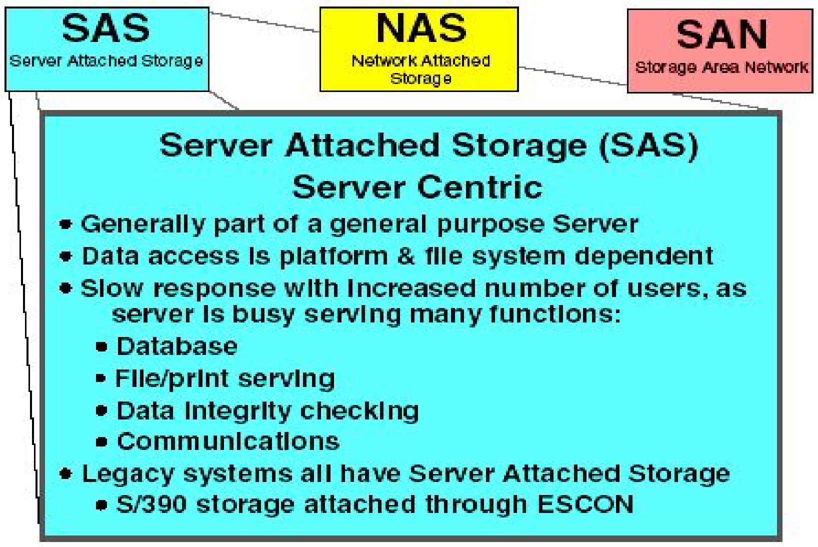

SAS
Server Attached Storage
NAS
Network Attached Storage
SAN
Storage Area Network
Network Attached Storage (NAS)
File-Centric
• Dedicated File Server
• File Server connected to a messaging protocol LAN
• Storage partioning
• Data copy-sharing between heterogeneous servers
• Data sharing between homogeneous servers

Figure 2.9: Network Attached Storage Overview [IBM2003c]

Server Attached Storage (SAS)

With server-attached storage a storage device is attached directly to the bus of a server [IBM2003c]. This approach results in a tight coupling between the storage device and server. Unlike NAS and SAN server attached storage attaches a dedicated storage device directly to a single server. Many legacy systems make use of server-attached storage.

Figure 2.10 lists some of the characteristics of server-attached storage systems.



SAS
Server Attached Storage

NAS
Network Attached Storage

SAN
Storage Area Network

Server Attached Storage (SAS)
Server Centric

- Generally part of a general purpose Server
- Data access is platform & file system dependent
- Slow response with increased number of users, as server is busy serving many functions:
  - Database
  - File/print serving
  - Data integrity checking
  - Communications
- Legacy systems all have Server Attached Storage
  - S/390 storage attached through ESCON

Figure 2.10: Server Attached Storage Overview [IBM2003c]

2.7 Common Data Centre Problems

Like any complex facility, data centres are prone to a number of problems. Some of these problems are often highlighted in material used to market virtualisation software. Seven major problems mentioned in the data centre literature will now be discussed.

Poor Resource Utilisation

This problem is mentioned frequently in material about virtualisation software, especially partitioning systems. According to [HP2004d] utilisation rates of HP-UX [HP2005c] users vary, but are typically around thirty percent. According to Cisco [CS2002a] Windows servers average twenty five percent utilisation. Sources such as Sun [SM2004c] and VMware [VMW2004b] put current server utilisation at between six percent and fifteen percent. Carolan and Radeztsky [CRS+2004] state that utilisation is *"often well below 25 percent to 30 percent"*. A study of six data centres collectively containing about 1000 servers by Andrzejak *et al.* [AAR2002] supports these claims. Although the numbers vary they are typically low. According to Sun

[SM2004g], utilisation rates are currently the lowest that they have ever been. This is a waste of resources as excess capacity is not being used. As a result the hardware is not being used to its full potential.

A data centre should be able to cope with resource outages, widely varying service demands and congestion. [HP2004a]. Data centre applications typically have dedicated resources assigned to them [SM2004b]. This "silo" effect results in applications being over-provisioned to meet peak demand [HP2005f]. A consequence of this is applications with fixed capacities and poor resource utilisation.

High Costs

According to Carolan *et al.* [CRS+2004] reducing costs is currently the highest priority for the data centre. Managing existing systems consumes around 69% of IT budgets [MS2004e]. The costs incurred to keep service at an acceptable level are very high, according to Cisco [CS2003a]. Many of these costs are a result of other problems highlighted in this section. The total amount of money spent on data centres looks set to rise further. According to [Sim2004] the market for data centre products and services is projected to rise by 47% by 2007.

Server Sprawl

Many organisations have deployed large numbers of servers in their data centres. This setup is difficult to manage and is referred to as server sprawl [PH2002]. Managing hundreds of servers is costly according to Sliwa and Vijayan [SV2002]. The large number of interdependencies between these systems compounds this problem [CRS+2004].

According to [PH2002] a number of factors contribute to server sprawl.

- Multitiered applications typically host different tiers on different servers.
- Separate servers used for development, testing and training to ensure isolation from the production systems.
- Separate servers for disaster recovery and equipment failures

Security Threats

The number of attacks against data centres is on the rise with new approaches constantly being devised to threaten the data centre [AP2004]. The tools used to launch these attacks are becoming increasingly sophisticated [AP2004]. Many of these tools are easy to use and can be downloaded from the Internet. To complicate matters further, there are also a number of new security issues to deal with, as a result of extranets being used by partners and suppliers, according to Cisco [CS2002a].

Difficulty Adapting to Changing Requirements

This refers to the difficulty of managing changing requirements in both the short and long term. Making changes to a data centre is time-consuming due to the large number of systems to be managed [SM2004g]. The diversity of these systems hampers the ability of an organisation to respond to opportunities [CS2003a]. In the short term this lack of flexibility results in data centres which are often unable to respond to fluctuations in service demand in a timely manner [CRS+2004].

Managing Multiple Operating Systems and Hardware Platforms

Data centres make use of multiple operating systems such as Unix (including many versions such as Solaris [SM2005b], HP-UX [HP2005c] and AIX [IBM2005e]), Windows [MS2005a] and Linux among others [CS2003a]. This results in compatibility issues and increased management overhead. In addition, mainframes, minicomputers and servers are commonly found in data centres [CS2003a].

Variety of Application Architectures

Data centres typically host a mix of monolithic mainframe applications, two-tier, three-tier and N-tier applications [CS2003a]. This is not to be confused with the previous point, which focuses on hardware and operating system heterogeneity rather than the architecture of applications.

2.8 Conclusion

Literature about data centres often focuses on physical and legal requirements such as fire suppression, cooling, power consumption and compliance with government regulations. Information about logical aspects of data centres such as applications, benefits and types of data centres is scarcer. Unlike in Chapter 3, the majority of information in this chapter is sourced from vendors such as Cisco and IBM. This is a result of the strong association between data centres and the major vendors.

A basic understanding of what data centres are and the applications which are hosted by these facilities is essential for contextualising the discussions in subsequent chapters. Chapter 5 presents a framework for evaluating server virtualisation systems. This discussion focuses on factors which are of relevance to commercial data centres.

Based on the classification by Cisco, two types of data centres were identified in this chapter. These two types, namely Internet data centres (specifically those of service providers) and enterprise data centres, are particularly relevant to the discussion of virtualisation applications presented in Chapter 3. These data centre types are also of relevance to the framework criteria presented in Chapter 5.

Although the overview of data centres presented in this Chapter is fairly high-level, it is sufficient for the purposes of introducing the data centre environment.

Chapter 3: Virtualisation

3.1 Introduction

An understanding of virtualisation is essential for effectively designing a framework for evaluating server virtualisation systems in order to compare them. The information presented in this chapter is used to inform the selection of criteria for this framework. This chapter is the second of three literature study chapters.

The recent increase in virtualisation research has been described as a "renaissance" by Figueiredo *et al.* [FDF2005]. Although this increase is relatively new, the principles on which this work is based is not. Virtualisation has been a subject of research for decades. The first virtual machine was designed and implemented by IBM in the mid 1960s by Adair *et al*. [ABC+1966]. This paper is referenced by one of its authors, R. J. Creasy, some fifteen years later in a historical discussion [Cre1981] of virtual machine work at IBM. This original virtual machine system was created for the IBM 360/40 [ABC+1966].

This chapter includes the following main sections:

- Virtualisation defined
- Virtual machines and server partitioning
- Approaches to virtualisation
- Advantages of server virtualisation
- Applications of virtualisation in data centres

3.2 Virtualisation Defined

According to Denning [Den2001] the term *virtual* originated from the field of optics before being adopted by Computer Scientists to describe a number of ideas ranging from virtual machines and virtual memory to virtual reality.

In this and subsequent chapters the focus will be on the term virtual as it applies to virtual machines and server partitioning systems. Smith and Nair [SN2005] provide an explanation of virtualisation:

"Virtualization provides a different interface and/or resources at the same level of abstraction" with the consequence that *"the real system is transformed so that it appears to be a different, virtual system or even a set of multiple virtual systems".* This is different from abstraction because abstraction is used to simplify an interface.

Another definition according to Singh [AS2004] is:

"Virtualisation is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others."

3.3 Virtual Machines and Server Partitioning

As stated in Chapter 1, the focus of this study is on virtualisation as it applies to server partitioning. There are various approaches to implementing server partitioning. These are discussed in greater detail in subsequent sections. Virtual machines and virtual machine monitors (VMMs) will be examined first.

Popek and Goldberg [PG1974] define a virtual machine as *"an efficient, isolated duplicate of the real machine".* Multiple duplicate machines can be hosted on a system using a virtual machine monitor (VMM). A VMM is defined by [PG1974] as a piece of software with three "essential characteristics":

1) *"provides an environment for programs which is essentially identical with the original machine"*
2) *"programs run in this environment show at worst only minor decreases in speed"*
3) *"the VMM is in complete control of system resources".*

Interestingly Popek and Goldberg [PG1974] exclude emulators and complete software interpreters from being classed as virtual machines due to the second requirement of efficiency. This classification is at odds with that of Smith and Nair [SN2005], which includes virtual machines which do not satisfy the efficiency requirement. Smith and Nair refer to Popek and Goldberg's requirements as those for *efficient* virtual machines. Their classification is outlined later in this section. Popek and Goldberg also exclude the *"usual timesharing operating system"* from being classified as a virtual machine due to the fact that it does not provide an identical environment.

A similar set of "essential characteristics" was also identified by Belpaire and Hsu [BH1975]:

- Logical equivalence between real machine and virtual machine environment
- The only performance degradation should be due to sharing of resources
- Virtual machines isolated by *"impassable walls"*.

In short a VMM is a layer of virtualising software used to host virtual machines. These virtual machines being hosted are referred to as guests. This is depicted in Figure 3.1.

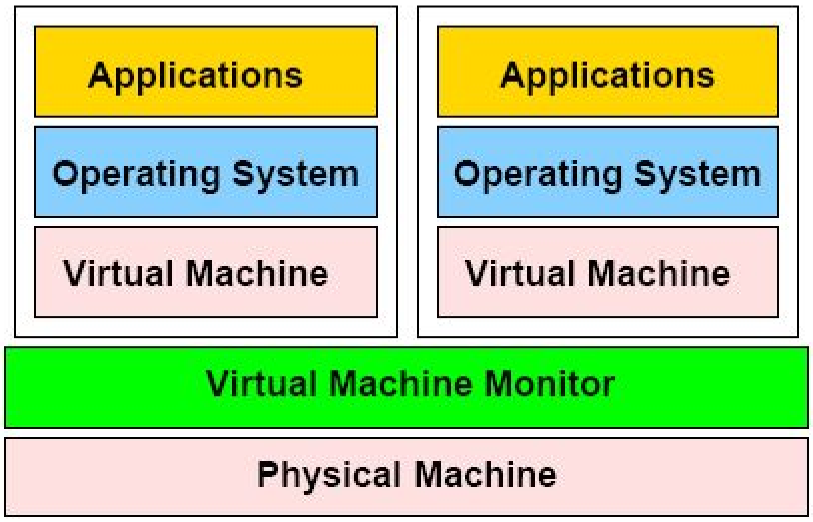


Figure 3.1: Virtual Machine Monitor Hosting Virtual Machines [SVL2001]

As can be seen in Figure 3.1 virtualisation can be used to make a single physical server appear as multiple virtual servers. The applications of this are discussed towards the end of this chapter.

According to Popek and Goldberg [PG1974] a virtual machine monitor consists of three components:

- **Dispatcher:** This component handles hardware traps and calls the other modules. System instructions trigger traps when executed in user mode.
- **Allocator:** As the VMM is in control of all system resources any requests for resources by guest operating systems need to be handled by the VMM. The allocator module is called by the dispatcher to perform this task when a guest issues a system instruction to allocate resources.
- **Interpreter:** If a guest issues a system instruction which is not used to allocate resources, the dispatcher calls an interpreter module instead of the allocator to simulate the instruction.

An interesting property of some virtual machines is recursiveness. Recursive virtual machines were examined by Belpaire and Hsu [BH1975] and defined as follows:

"If the VMM can run on a Virtual Machine to generate another level of VMs, the system is a Recursive Virtual Machine System. Its structure is a potentially infinite tree of VMs.".

This property enables virtual machines to be layered above other layers of virtual machines.

Virtual machines are divided into two main categories by Smith and Nair [SN2005], namely process virtual machines and system virtual machines. Process virtual machines such as high-level language virtual machines and binary optimisers are created to support a single process. System virtual machines provide entire virtual systems which can host operating system instances. The focus of this study will be on system virtual machines because these are used for server partitioning. Process virtual machines such as the Java Virtual Machine (JVM) are beyond the scope of this discussion.

The different types of virtualisation systems which are used for server partitioning are covered in the next section. Figure 3.2 provides a high-level taxonomy of virtual machines. Instruction set architectures (ISA) feature prominently in this taxonomy.

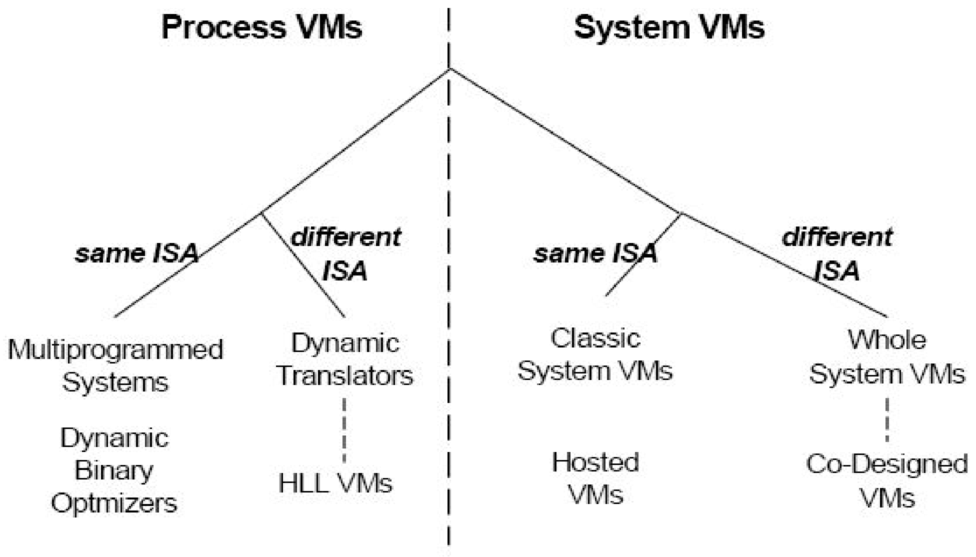


Figure 3.2: A Virtual Machine Taxonomy [SN2005]

3.4 Approaches to Virtualisation

As mentioned previously the focus of the framework which is the subject of this research is server partitioning systems. For the remainder of this document the term virtualisation will be used to refer to server partitioning systems.

A number of approaches have been taken to implementing this type of virtualisation. Some of these approaches, such as physical partitioning systems and containers, are not generally referred to as virtual machines, as can be gathered from Tucker and Comay [TC2004]. Interestingly, Smith and Nair [SN2005] use the term virtual machine monitor when discussing physical partitioning systems. The different partitioning techniques are arranged in a taxonomy by Smith and Nair [SN2005]. This is depicted in Figure 3.3. Each of these approaches will be examined in the sections which follow.

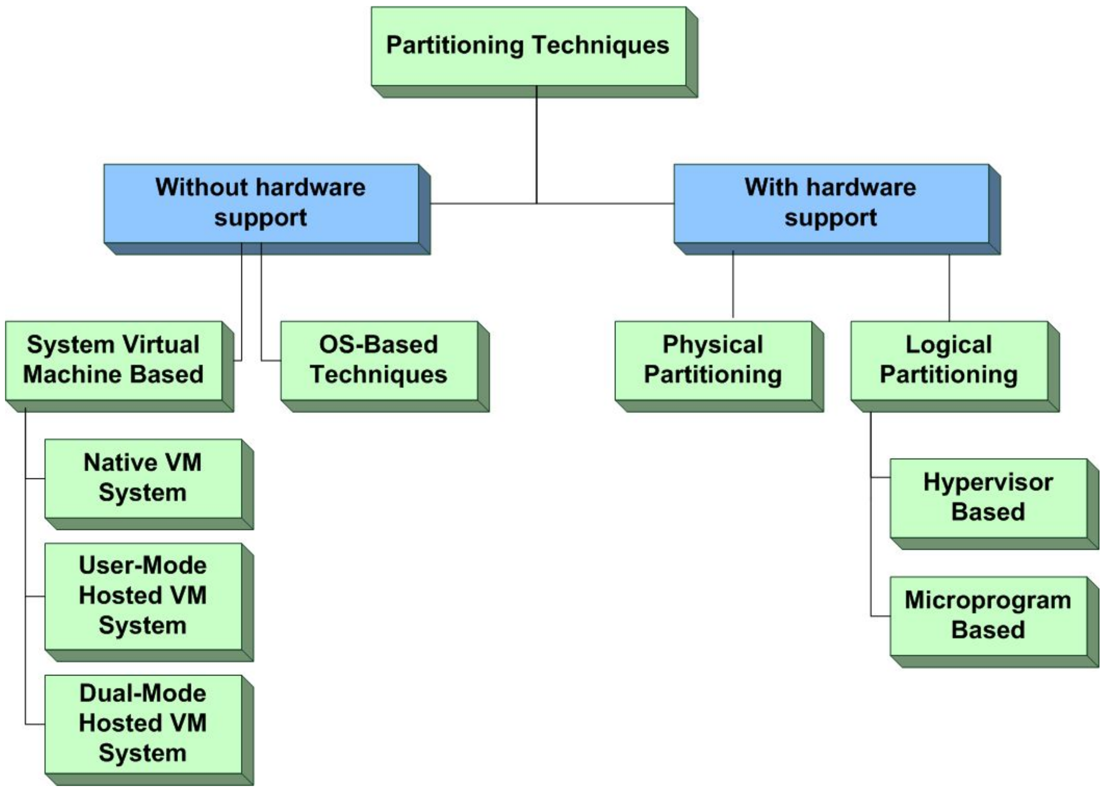


Figure 3.3: A Taxonomy of Partitioning Techniques (adapted from [SN2005])

System Virtual Machines

System virtual machines can be divided into three categories based on the implementation approach employed. This classification is discussed by Robin and Irvine [RI2000] and King *et al.* [KDC2003] based on earlier work by Goldberg [Gol1972] cited by both these sources. This classification is also discussed by Smith and Nair [SN2005], using slightly different terminology. This terminology will be adopted for this document as the author believes that it is the most descriptive. The three categories are:

- Native virtual machine monitors

- User-mode hosted virtual machine monitors
- Dual-mode hosted virtual machine monitors

These three types of virtual machine systems will now be discussed.

Native Virtual Machine Monitor

This is also referred to as a Type I VMM [KDC2003]. With this approach the VMM forms a virtualisation layer directly between the underlying hardware and guest operating system instances. The virtual machine monitor executes in a higher privilege mode than any other software on the system [SN2005]. This often results in the supervisor mode of guest operating systems being simulated in software by the VMM. There is no host operating system between the hardware and the virtual machine monitor. The VMM is responsible for resource management and scheduling of guests [RI2000] This improves performance, but increases the complexity of the implementation. A native virtual machine monitor cannot take advantage of the I/O facilities provided by a host operating system. A native virtual machine monitor is depicted in Figure 3.4.

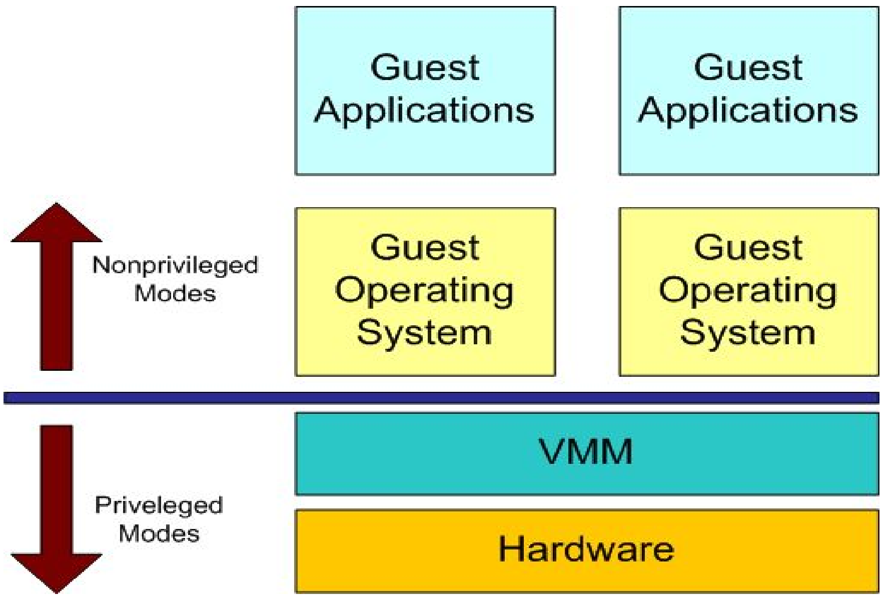


Figure 3.4: A Native Virtual Machine Monitor (based on figures in [KC2002] *and* [SN2005]*)*

User-Mode Hosted Virtual Machine Monitor

A second, but less efficient approach is to host the VMM entirely on a host operating system. This is simpler as the VMM can take advantage of the services offered by the host operating system. This is also known as a Type II VMM [KDC2003]. Each guest typically executes as a process on the host. A Type II VMM is depicted in Figure 3.5.

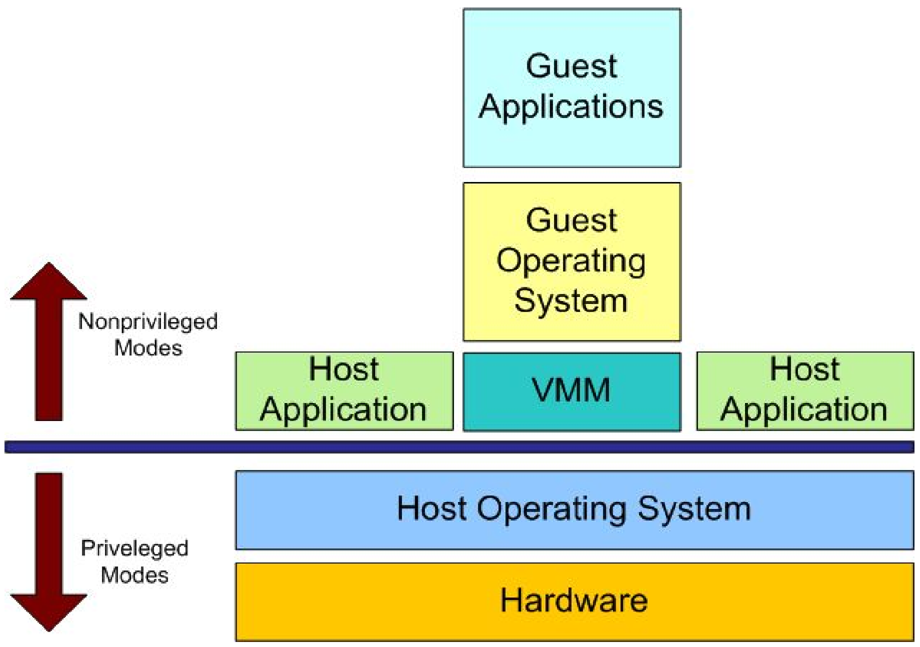


Figure 3.5: User-Mode Hosted Virtual Machine Monitor (based on figures in [KC2002] *and* [SN2005]*)*

Examples of this type of VMM include User Mode Linux (UML) [UML2005] and UMLinux [UM2005]. For these systems the kernel is ported to the system call interface of a host operating system [Dik2001a]. Höxer *et al.* [HBS2002] examined the issues encountered when implementing this type of system. The most common issue is dealing with system calls from applications running on a hosted kernel. To prevent these calls from executing in the host kernel, a trace process is used to intercept system calls and redirect them to the hosted kernel.

Memory for each virtual machine is usually implemented using a memory mapped file on the host. The size of this file is the same as the quantity of physical memory the virtual machine believes it has. Interrupts and exceptions also need to be emulated. Hardware devices are virtual, using functionality provided by the host [Dik2000].

These implementation issues result in significant overhead. This is well known and documented by Barham *et al.* [BDF+2003] and Surányi *et al.* [SHH+2005]. The overhead of context switching between processes is the most significant performance issue [Dik2001b]. According to this source, UML was created to facilitate kernel development, but has other applications including virtual web hosting. The benefits to kernel developers far outweigh the performance overhead. For data centres hosting web sites this overhead may be an issue.

This approach is not compatible with all existing applications. According to Dike [Dik2001b], emulators and certain types of installation programs are not compatible with UML.

The security of a Type II (user-mode) VMM is also dependent on the security of the host operating system [RI2000]. A weakness in the host operating system could be exploited to compromise the security of the VMM.

Dual-Mode Hosted Virtual Machine Monitor

A dual-mode virtual machine monitor is hosted partially in nonpriveleged mode and partially in privileged mode. This is also referred to as a hybrid VMM [KDC2003]. According to this source a hybrid VMM accesses the hardware directly, but relies on a host operating system support for I/O. Switching between the VMM and the host operating system during I/O operations results in increased CPU overhead [SVL2001]. The major advantage of this type of VMM is that it can use the device drivers provided by the host operating system [KC2002].

A dual-mode hosted VMM consists of three components [SN2005]:

- **VMM-n (native):** This component executes directly on hardware and is tasked with handling guest operating systems' attempts to directly execute privileged instructions. It may also be used to patch sensitive instructions which do not trigger traps. This is discussed further in the section on processor virtualisation.

- **VMM-u (user):** This user-mode process makes use of the host operating system to perform I/O and allocate other resources.

- **VMM-d (driver):** This component is installed in the host as a device driver. This provides a communications channel between VMM-u and VMM-n.

VMware Workstation [SVL2001] and Microsoft Virtual Server 2005 [MS2004c] are examples of this type of VMM. Figure 3.6 depicts a dual-mode virtual machine monitor.

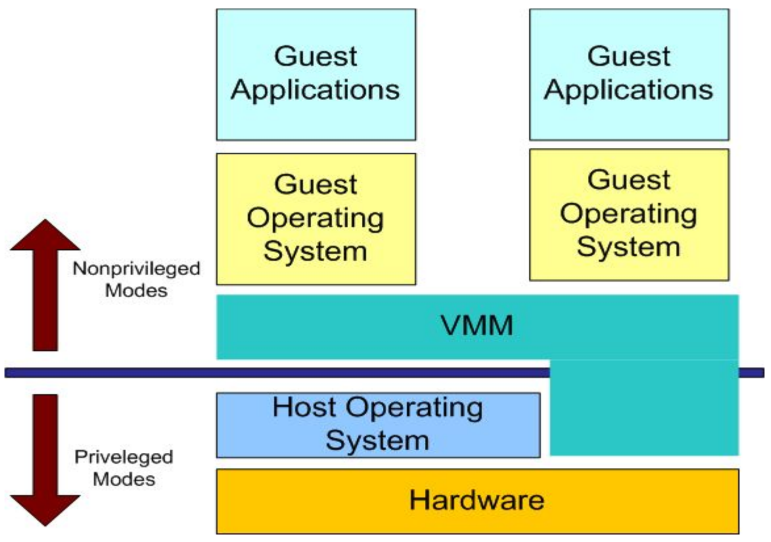


Figure 3.6: A Dual-Mode Hosted Virtual Machine Monitor (based on figure in [SN2005])

Resource Virtualisation (Including Paravirtualisation and Other Optimisations)

One of the properties of virtual machine monitors identified by Popek and Goldberg [PG1974] was that of resource control. The VMM must control all access to resources. These resources need to be virtualised so that the VMM can ensure integrity and compatibility. The approaches to virtualising different types of resources will be examined shortly.

A recent trend in virtualisation which is relevant to this discussion is paravirtualisation. This term was proposed by the creators of Denali [WSG2002a] and later adopted by the creators of Xen [BDF+2003]. Paravirtualisation involves modifying an architecture in order to make it more suitable for virtualisation. Guest operating systems are “aware” that they are hosted by a VMM. Denali and Xen sacrifice compatibility with existing operating systems in the name of performance. As demonstrated by Xen this does not have to come at the cost of incompatibility with existing applications. As noted in the Xen paper [BDF+2003], only a simple operating system was originally ported to Denali. Since then a full operating system (NetBSD) has been ported [WCS+2004]. Paravirtualisation is discussed further under each of the resource types.

Smith and Nair [SN2005] note that handshaking, a concept in use since the 1970s in the IBM System/370, is similar to paravirtualisation in the sense that it also involves modifying guest operating systems in order to improve performance. Handshaking enables a number of optimisations by allowing the VMM and guest operating system instances to communicate. This improves coordination between them and can be used to reduce the duplication of functionality between guest and VMM. Providing the VMM with access to the workings of guest operating systems is similarly beneficial.

Processor Virtualisation

Popek and Goldberg [PG1974] formally defined a set of requirements that an instruction set architecture (ISA) must satisfy in order to be virtualisable. Their findings are presented in the form of theorems. The main theorem (Theorem 1) of this paper contains a number of terms which will be defined first:

- A *privileged instruction* is defined as one which traps when executed in user mode, but not in system mode.
- A *control sensitive* instruction is an instruction which attempts to alter the privilege level or resource allocation of a virtual machine.
- A *behaviour sensitive* instruction is one whose behaviour depends on the privilege level in which it is executed or its location in memory.
- A *sensitive instruction* is an instruction which is either control sensitive or behaviour sensitive.

Popek and Goldberg's [PG1974] main theorem is as follows:

"For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions."

This means that an ISA is not virtualisable if it contains sensitive instructions which do not trap (can be executed in user mode).

It is worth mentioning that ISAs which do not meet this requirement can be, and have, been virtualised using less elegant approaches. Consider the example of the x86 ISA. It is well known that the x86 architecture does not meet the formal requirements for virtualisability. This was examined in detail by Robin and Irvine [RI2000]. The Pentium instruction set exposes sensitive instructions to non-privileged users. This compromises the isolation, and therefore security, of virtual machines. Virtual machine monitors for x86 such as VMware [Wal2002] currently employ binary rewriting (translation) of instructions to work around this problem.

Paravirtualisation systems such as Xen [BDF+2003] modify the guest operating systems to execute in a lower privilege mode. The modified Xen architecture also differs from the x86 architecture in that the need for binary translation of sensitive instructions which do not trap is removed. In order to boost performance a fast handler for system calls is installed by guest operating systems to prevent system calls made by guest applications from being received via the VMM. A similar mechanism for page faults was not possible due to privilege restrictions.

Clearly paravirtualisation systems which require guest operating systems to be modified do not meet Popek and Goldberg's [PG1974] equivalence requirement for virtual machine monitors.

Another recent development is Intel's VT-x (Vanderpool) technology which, like AMD's Pacifica [AMD2005] is designed to simplify the task of implementing VMMs for x86 and improve performance. VT-x is discussed by Uhlig *et al.* [UNR+2005]. These provide hardware support for processor virtualisation with features such as additional privilege levels and new instructions for transitions between the VMM and guests. Sensitive instructions which did not result in traps now trap when executed by guests. The idea of modifying the hardware to provide improved virtualisation support is not new. Smith and Nair [SN2005] discuss hardware assists which were added to the IBM System/370 to improve performance:

Memory Virtualisation

The approach taken to implementing memory virtualisation depends on the architecture. This is discussed by Smith and Nair [SN2005] and Barham *et al.* [BDF+2003]. For this discussion some understanding of virtual memory concepts is assumed, although some basic terms will briefly be explained.

A **page table** is a data structure used to map virtual pages to real pages. Each guest operating system maintains a page table for each application it hosts. When an operating system is executing on bare hardware these real pages correspond to pages in physical (machine) memory.

The **translation lookaside buffer** (**TLB**) is a cache used to speed up lookups in the page table.

A **TLB miss** occurs when an attempt is made to access a page whose mapping is not cached by the TLB.

In some architectures the page table is architected, while in others the translation lookaside buffer (TLB) is architected. If the *page table is architected,* it means that page table management instructions are built into the instruction set architecture (ISA). In the event of a TLB miss, a traversal of the page table occurs in hardware to locate the appropriate key-value pair. This mapping is then cached by the TLB. The x86 and IBM System/370 ISAs are examples of page table architected architectures. In these types of systems memory is typically virtualised using *shadow page tables*. The virtual pages of the guests are mapped to real pages using the pages tables of the guest operating systems. The role of the shadow page tables, which are managed by the VMM, is to map these real pages to physical (machine) pages. This extra level of mapping results in a performance overhead for this type of system. Waldspurger [Wal2002] and Rosenblum and Garfinkel [RG2005] discuss the use of shadow page tables in VMware.

If the *TLB is architected,* there are instructions defined in the ISA for managing the TLB. For these architectures a TLB miss results in a trap into the operating system to find the appropriate page table entry. According to Barham *et al*. [BDF+2003] the TLB is architected in the Alpha, MIPS and SPARC ISAs. In order to virtualise these systems the TLB needs to be virtualised. One approach is to copy the TLB of a guest operating system over the physical TLB when it becomes active. Of course the real pages will first have to be translated to physical (machine) pages before the copy. This approach results in significant overhead. An alternative approach is to use the address space identifiers (ASIDs) of the TLB. Each entry in the TLB includes an ASID to associate different entries with different applications (address spaces). These can be used to identify different guest operating system address spaces allowing the TLB to be shared between all guests.

For more information on these memory virtualisation techniques refer to [SN2005].

According to Barham *et al*. [BDF+2003] keeping shadow page tables up to date results in significant overhead for update-intensive tasks. Paravirtualisation can improve the performance of memory virtualisation. Xen requires guest operating systems to be modified to only access memory pages allocated to them by Xen VMM. These pages may be discontiguous. All updates to the page table are validated by the Xen VMM. Performance is improved because shadow page tables do not need to be maintained and memory can be accessed without a second level of mapping (indirection). Despite this improvement validating page table updates results in some performance overhead.

Similarly handshaking, which was introduced at start of this section, can also be used to eliminate the use of shadow page tables by using **nonpaged mode**. Operating systems hosted by the IBM System/370 can define a real address space which is as large as the largest virtual address space it will require. The guest disables dynamic address translation so only one level of mapping takes place.

Pseudo-page-fault handling, which is also discussed by Smith and Nair [SN2005], is another handshaking optimisation. When a page fault occurs due to the VMM having swapped the pages of one virtual machine out for another, the operating system is notified that a special type of page fault has occurred. This enables the guest operating system to schedule another process, while the fault is being handled by the VMM.

Other optimisations are examined in [SN2005].

Intel's VT-x and AMD's Pacifica which were discussed under processor virtualisation also provide assists for memory virtualisation. For more information refer to a recent presentation by AMD [AMD2005].

Input/Output (I/O) Virtualisation

One of the challenges facing those implementing virtualisation systems is supporting a wide range of hardware. Hardware vendors such as Sun [SM2004d], Hewlett Packard [HP2004e] and IBM [IBM2004b] only have to support a relatively limited set of hardware when virtualising non-x86 systems. Virtualisation systems for x86 such as VMware [SVL2001], Xen [FHN+2004] and Microsoft Virtual Server [MS2004c] have to support many more devices. The compatibility advantage of a user-mode hosted or dual-mode hosted VMM is that the drivers of a host operating system can be used by the VMM, at the cost of performance overhead. This approach is discussed by Sugerman *et al*. [SVL2001] and Rosenblum and Garfinkel [RG2005]. These types of VMMs were discussed earlier in this chapter. Native virtual machine monitors such as VMware ESX Server [Wal2002] usually need to include device drivers for the devices they support. An alternative approach is to access devices via a dedicated I/O virtual machine. IBM Logical Partitions (LPARs) for pSeries servers [IBM2005b], which are examined further in Chapter 4, make use of a dedicated I/O Server virtual machine. This was recently implemented for Xen [FHN+2004]. An important advantage of this approach is the isolation of virtual machines from being compromised by device driver failures which would otherwise have executed in privileged mode as part of the virtual machine monitor.

Smith and Nair [SN2005] identified five ways in which a device can be virtualised:

- **Dedicated devices**: Each device is assigned to a single partition.
- **Partitioned devices**: Devices such as disks can be partitioned into multiple logical devices.
- **Shared devices**: Multiple virtual machines have access to the same device
- **Spooled devices**: Spooled devices such as printers write their output to a buffer which is processed by the device. This is similar to sharing except that it takes place on a much coarser timescale.

- **Nonexistent devices**: Virtual machine monitors can provide a virtual device to a virtual machine, even if the device does not exist physically. A virtual network adapter linking virtual machines is a classic example.

Smith and Nair [SN2005] discuss three levels at which a device can be virtualised:

- **I/O Operation Level**: I/O operations, whether implemented as privileged instructions such as on mainframe and x86 systems, or by memory mapping on RISC platforms, represent a low-level interface at which devices can be virtualised. Each time I/O takes place a number of I/O operations are performed. In order to virtualise a device at this level, the VMM has to examine the stream of I/O operations to determine the sequence of actions they represent.
- **Device Driver Level**: The device driver interface is another level at which a device can be virtualised. This involves providing a virtual device driver for each device type to be virtualised. This device driver needs to be ported to each guest operating system which is to be supported. Calls to the interface of this virtual device are translated to calls to the driver interface used by the VMM.
- **System Call Level**: A high-level approach to virtualising devices is to implement this at the system call interface. System calls, such as *read,* need to be emulated for each operating system supported.

Resource Scheduling and Guarantees

Virtualisation is also useful for insulating applications in one partition from the resource demands of other applications. Virtual machine monitors have control over system resources and can therefore enforce resource guarantees. These types of guarantees are not exclusively the domain of system VMMs. Other types of virtualisation systems such as operating system containers, which will be discussed in the next section, and logical partitions, can also enforce these types of guarantees. This is also useful against denial of service attacks [AG2001]. Resources such as CPU processing capacity, memory, disk bandwidth, disk space and network bandwidth can be allocated to partitions. This is particularly important to service providers who wish to assign resources to customer Web sites based on the relative importance of customers according to Sullivan and Seltzer [SS2000]. This will be discussed further in Chapter 5.

Resources can be managed by operating systems and system virtual machines using scheduling techniques. Schedulers exist for all of the main resources types, including disk bandwidth, network bandwidth and processor capacity. Examples of these types of schedulers are cited in Chapter 5, and their applicability to virtualisation systems is discussed.

A detailed study of a wide range of schedulers and their implementation details is beyond the scope of this study. Lottery and stride scheduling, which were proposed by Waldspurger [Wal1995], are discussed next as an illustrative example.

Waldspurger discussed these scheduling algorithms in the context of other scheduling schemes used by operating systems. The most common approach used by operating systems is priority based scheduling. In order to control the relative rate at which tasks are executed alternative scheduling algorithms are required. Fair share scheduling and proportional share scheduling algorithms are two well known approaches. Fair share schedulers aim to control resource allocation averages over a period of time. These schedulers are implemented by varying process priorities in order to achieve its goals.

Proportional share schedulers allocate resources on a much finer time scale. Examples of proportional share scheduling algorithms are stride scheduling and lottery scheduling. These were both proposed by Waldspurger [Wal1995]. With lottery scheduling each process is assigned a number of tickets. Resources are assigned to the task which was assigned the “winning” lottery ticket. A winning ticket is periodically chosen by random selection. More important tasks are assigned more tickets, and thus have a higher probability of “winning” the lottery. Stride based scheduling involves calculating how often each task should be assigned resources. This interval is called a stride.

In order to improve resource utilisation while maintaining resource guarantees Sullivan and Seltzer [SS2000] extended the lottery scheduling approach by adding ticket exchanges. Each task is assigned separate tickets for each type of resource. Consider two tasks which each have tickets for resources that the other requires. If one task has excess CPU tickets while the other has excess memory tickets the two can exchange tickets. This does not affect the proportion of tickets assigned to other tasks. This is illustrated in Figure 3.7. Tasks A and B exchange fifty tickets for each resource type in this example.

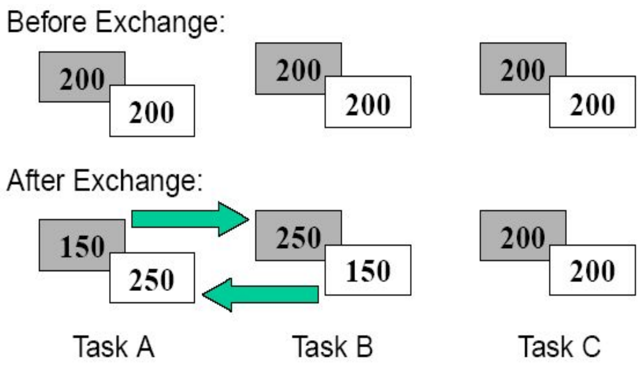


Figure 3.7: Ticket Exchange [SS2000]

These types of schedulers can be used in conjunction with resource containers or system virtual machines to provide resource guarantees.

Operating System-Based Partitioning – Containers

An alternative approach to using a classic system virtual machine to partition a server is to create a virtual execution environment within a single operating system instance. For the remainder of this document the term *container* will be used to generically refer to this type of partitioning system. The term container was adopted because this term, unlike others, has been used by more than one source. The term "resource container" is used by Banga *et al*. [BDM1999] to refer to a system used to logically group resources. Sun uses the term "Solaris Container" [SM2004e] to refer to their OS-based partitioning system.

Containers provide varying degrees of isolation within a single operating system instance. Systems like FreeBSD jails [KW2000], Linux VServer [LVS2005, BL2005], Virtuozzo [Vir2005] and Solaris Containers [SM2004e, TC2004] provide an isolated environment to users which appears similar to a separate server. The degree of similarity and types of isolation provided depend on the system. For more information on the different types of isolation refer to Chapter 5.

Before discussing the container-based approach to partitioning servers, the relevance of *resource* containers to this type of partitioning is discussed. Although *resource* containers [BDM1999] do not provide virtualisation functionality the resource management concepts outlined by this source are essential for providing resource isolation for container-based systems. For this reason *resource* containers will be discussed next.

A resource container is defined by [BDM1999] as *"an abstract operating system entity that logically contains all the system resources being used by an application to achieve a particular activity"*. This approach overcomes a number of shortcomings of general-purpose operating systems identified by this source. In these operating systems processes are treated as independent activities. This is contrary to the behaviour of many server applications which often create multiple processes. An example of this is an HTTP server which creates one process per connection. This is depicted in Figure 3.8.

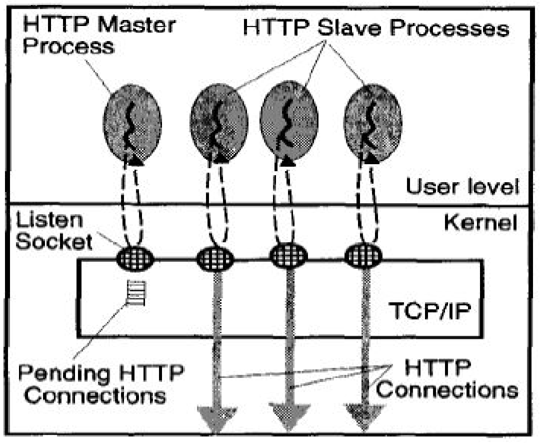


Figure 3.8: One Process Per Connection [BDM1999]

Many processes are used to service client requests. Service providers often wish to provide prioritised service to certain clients. It is incorrect to treat each process as a completely independent entity. Another common scenario mentioned by [BDM1999] occurs when a single process (or one process per CPU) handles all requests. This occurs when a one thread per client approach is used as can be seen in Figure 3.9 or an event driven approach as shown in Figure 3.10.

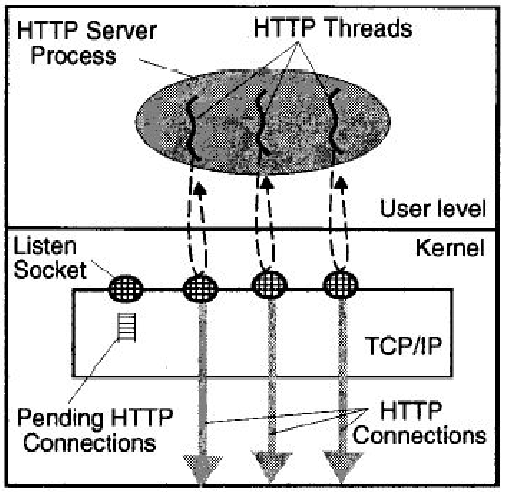


Figure 3.9: One Thread Per Connection [BDM1999]

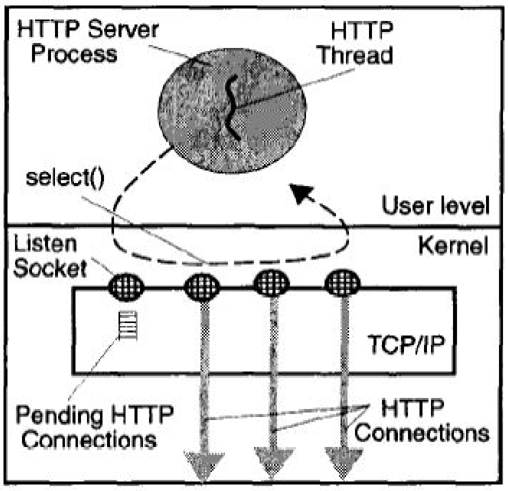


Figure 3.10: One Process for All Connections [BDM1999]

Another issue with such operating systems is that processing carried out by the kernel is often not counted as resources consumed by a process. This can happen in interrupt driven applications such as networking. Alternatively this processing is accounted as that of the process which is active at the time of the interrupt. This hidden processing is depicted below the line in Figure 3.11.

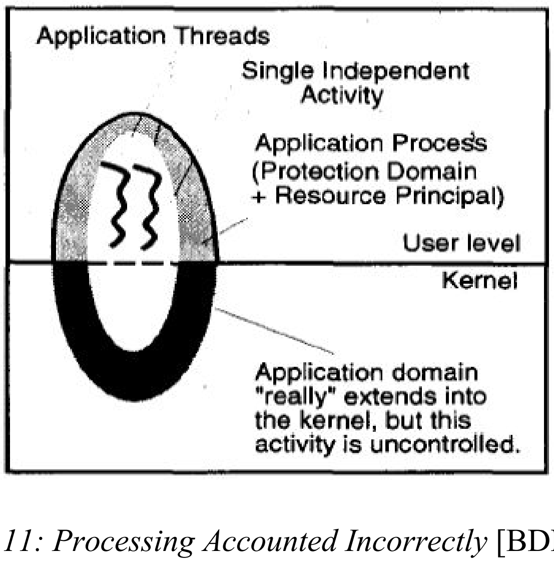

Application Threads
Single Independent Activity
Application Process (Protection Domain + Resource Principal)
User level
Kernel
Application domain "really" extends into the kernel, but this activity is uncontrolled.

Figure 3.11: Processing Accounted Incorrectly [BDM1999]

Resource containers can be used to overcome these resource management challenges. These containers group related activities. This together with accurate resource usage accounting can significantly improve the resource management facilities offered by an operating system. Resources can be assigned to containers rather than individual processes.

Allocating and managing resources is only one aspect of a container-based system. For the remainder of this document only container-based systems providing virtualisation features will be considered. Container-based virtualisation systems (to varying degrees) present the user with what *appears* to be multiple virtual servers. These are in fact containers which may share the kernel with many other containers. Securing and isolating these containers from each other is an important goal, as is providing the illusion of a separate operating system instance. According to Kamp and Watson [KW2000] this approach is compatible with virtually all applications. Each container has its own root password, IP address and a subset of the original filesystem. Processes in one container cannot access information about processes in other containers.

In order to implement this functionality significant changes usually have made to the underlying operating system. New security restrictions have to be enforced to prevent the security of containers from being compromised. For example only a subset of the process table is visible to each container. An alternative approach to making these changes to the operating system is to implement this functionality in user space. This was successfully implemented by Surányi *et al.* [SHH+2005]. Security provided by these "pot spaces" is claimed to be equivalent to that of a FreeBSD jail [KW2000]. Like jails, this approach also has very little overhead. Solaris Containers, a term used to refer to Zones which make use of resource management, are also claimed to have little or no overhead [TC2004]. These containers can be rebooted individually and have resources assigned to them. Solaris Containers are examined in greater detail in Chapters 4, 5, 6 and 7.

Containers result in lower performance overheads compared to other approaches. Fewer resources are consumed by containers than multiple operating system instances. System virtual machines make use of the latter approach and consume more resources as a result.

A container-based approach does not have to deal with many of the challenges facing system virtual machine-based systems such as architectures which are not virtualisable. Virtualisation takes place at the system call level. The main disadvantage of containers, is that a single kernel is shared between the containers. If one container triggers a crash in the kernel, all the other containers will be affected. Thus the stability of the kernel is an important consideration when opting for this approach.

Interestingly the merits of virtual machines versus virtualising the operating system were considered by IBM in the early 1970s [BH1973]. The CMS operating system in use at the time did not support multiple users. A virtual machine providing multiple CMS instances was considered superior to modifying CMS for multi-user access because CMS did not provide support for resource sharing and other protection mechanisms. Although the "virtual operating system" idea discussed was that of the classic process virtual machine there are parallels between this approach and the container-based systems of today. Both require operating system modifications and share a single OS instance, providing a lower level of isolation than a classic system virtual machine.

Physical Partitioning

Physical partitioning is a hardware-based approach to server partitioning. Each partition has dedicated physical resources assigned to it. This is not to be confused with virtualising hardware using a virtual machine monitor or with containers. Partitions are physically isolated from each other, providing a higher degree of isolation than any other virtualisation approach. Hardware failures in one partition do not affect other partitions [HP2004b]. Complete software isolation is another benefit of this approach. Each partition hosts a separate operating system instance. These operating systems do not have to be the same version, or in some cases even the same operating system.

Resources are assigned at a coarse level of granularity. Typically one or more system boards containing multiple CPUs and gigabytes of memory are assigned to each partition. Figure 3.12 depicts a server which has been physically partitioned. System boards are connected via a high speed interconnect.

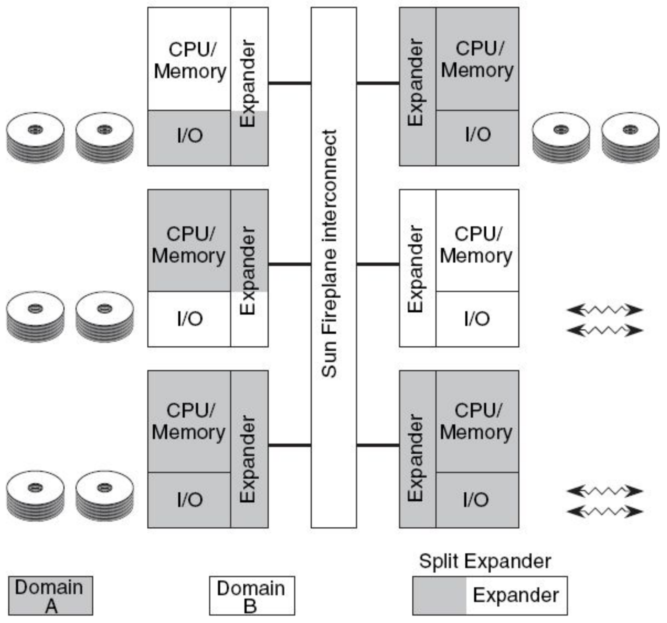


Figure 3.12: A Sun Server Physically Partitioned into Two Domains [SM2003b]

Some physical partitioning systems support dynamic resource reallocation. Resources such such as memory and CPUs can be migrated from one partition to another without rebooting any of the partitions.

Physical partitioning is currently restricted to larger systems. Support for this type of virtualisation needs to be built into the server. Hewlett Packard Node Partitions (nPars) [HP2004i] and Sun Dynamic System Domains [SM2004f] are examples of this type of virtualisation system. For more information on these systems refer to Chapters 4 and 6. High-end server models such as the HP Superdome and the Sun Fire E25K provide physical partitioning functionality.

Logical Partitioning

Logical partitions are similar to system virtual machines. Multiple operating system instances are hosted on a single server. As with system VMs the underlying hardware resources of different partitions are not physically isolated.

For logical partitioning systems the virtualisation layer between the hardware and the partitions is referred to as a *hypervisor*. This corresponds to the virtual machine monitor of a system virtual machine system. Smith and Nair [SN2005] make a distinction between logical partitioning and system virtual machines. A hypervisor requires hardware support and executes in a *"special mode"*. This mode enables the hypervisor to execute in a higher privilege mode than the guest operating systems without the guests having to execute in user mode. Hardware support includes features such as replication of registers to support context switches and registers to store the state of the hypervisor. Hypervisor state registers can include registers to store the lower and upper memory address bounds of each partition.

Interestingly the Xen [BDF+2003] project has adopted the term hypervisor to refer to their virtual machine monitor. This is despite the fact that the underlying hardware does not support any special privilege mode (not without VT-x [UNR+2005] or Pacifica [AMD2005] support). Xen makes use of three of the four x86 privilege levels (rings). Xen itself is hosted in ring 0 (most privileged). Guest operating systems, which would normally execute in ring 0, are ported to ring 1. User applications of these guests execute in ring 3 (least privileged) as normal. This approach enables Xen to execute in a more privileged level without resorting to hosting the guest operating systems in the least privileged mode alongside user applications. Ring 0 can hardly be considered a special mode designed for hypervisors.

It could be argued that any x86 VMM which is enhanced to take advantage of the new privilege modes provided by Intel VT-x [UNR+2005] or AMD Pacifica [AMD2005] could be referred to as a hypervisor.

Older logically partitioned systems relied on microcoded logical partitioning instructions whereas newer systems make use of a standard ISA, supplemented with a "special" mode as discussed previously. For more information on this topic refer to [SN2005]. IBM Logical Partitions (LPARs), which take the newer approach, will be examined in Chapters 4 and 6.

As noted by Smith and Nair [SN2005] the reliance of logical partitioning systems on hardware support prevents them from hosting other layers of logical partitions recursively.

3.5 Advantages of Server Virtualisation

The advantages of server virtualisation put forward by researchers working on this technology are numerous. Some of these advantages such as using virtual machines to facilitate operating system debugging and development are not relevant to commercial data centre environments. Only advantages which are relevant to data centres will be discussed here. The most significant of these advantages are identified and presented next.

Improved Server Utilisation

One of the common data centre problems identified in Chapter 2 was that of poor server utilisation. This was confirmed by the work of Andrzejak *et al.* [AAR2002]. When multiple applications are hosted on the same server the resources of that server are pooled. Smith and Nair [SN2005] also list improved system utilisation as one of the benefits of virtualisation.

With many server virtualisation systems, resources can be reassigned dynamically based on application demand or other performance goals. Those promoting the use of virtualisation emphasise that this results in improved utilisation. As mentioned in Chapter 2, utilisation rates of servers in data centres are currently low. Server virtualisation enables multiple applications each running on a separate underutilised server to be consolidated onto a single server [MS2004a]. This results in improved utilisation. This is clearly illustrated in Figure 3.13.

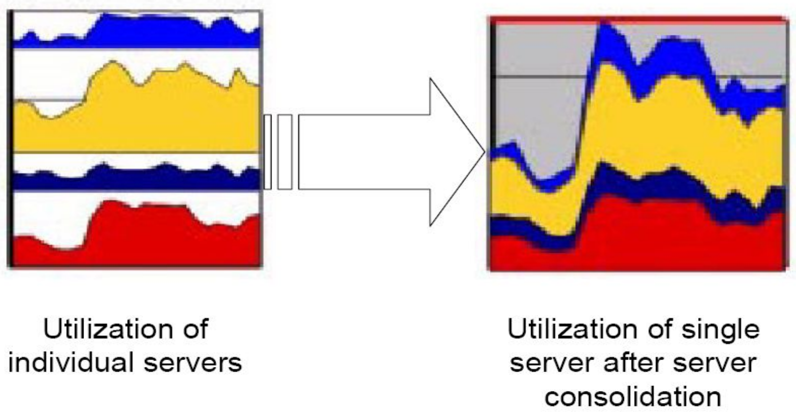


Figure 3.13: Consolidation and Server Utilisation [IBM2004c]

Isolation of Settings

Applications with conflicting settings can also be placed in separate partitions. In this way conflicts resulting from installation of new applications are avoided. Application installation settings need to be isolated according to Jiang *et al.* [JXE2004]. Virtualisation provides this type of isolation. Port conflicts, another common problem, can also be avoided [PT2004]. Virtualisation also enables different versions of the same application to safely coexist on the same server simultaneously. Partitioning can also be used to isolate operating system settings such as kernel tuning parameters.

Reduced Administration and Hardware Costs

Manageability is improved as applications with conflicting settings can now be safely installed on the same system. Utilisation improvements are also claimed to result in reduced hardware expenditure [VMW2004c]. If hardware can be better utilized, less equipment needs to be purchased. This also results in a smaller number of servers to manage [HP2002]. Fewer servers are easier to manage and administrative costs are reduced [HP2004d] as a result. This corresponds to the problem of server sprawl identified in Chapter 2. Server partitioning enables administrative rights for partitions to be safely delegated to different groups according to Kamp and Watson [KW2000] and Price and Tucker [PT2004].

Support for Multiple Time-Zones on a Single Server

Server virtualisation systems such as those examined in Chapter 4 enable each partition to be set to a different time zone. This is especially useful for servers which provide services to different regions in different time zones. Using virtualisation, downtime can be scheduled for each geography separately, even though these systems share the same physical server [SN2005].

Simpler Enforcement of Quality of Service (QoS) Guarantees

Virtualisation enables administrators to cleanly separate workloads into isolated partitions. This is an improvement over the traditional approach of allocating resources to individual processes. Banga *et al.* [BDM1999] explored the concept of resource containers. The difficulties of using a traditional operating system to enforce resource guarantees were explored in the section on container-based systems. A virtual machine monitor is also able to manage resource allocations, because according to Popek and Goldberg [PG1974] the VMM must be in control of the system resources. This can be used to implement quality of service (QoS) guarantees.

Improved Security

Security isolation is very important to service providers who host websites for different customers securely on the same server. One web application should not be able to access the data or applications belonging to other customers. The growing number of security threats facing data centres was discussed in Chapter 2. Belpaire and Hsu [BH1975] identified the need to isolate virtual machines from one another as one of the essential characteristics of virtual machines. This and other characteristics were discussed earlier in this chapter.

Improved Fault Isolation

The application of virtualisation to isolate faults was investigated by Buzen *et al.* [BCG1973]. Faults can be localised and isolated to a single partition. This results in improved system availability. An application running within a software partition is able to survive software faults which may occur in other partitions [HP2002]. Similarly, applications running within physical partitions can operate unaffected by hardware failures in other hard partitions [HP2002].

Reduced Downtime

A logical result of fault isolation is a reduction in server downtime. Clark *et al.* [CFH+2005] discuss the migration of virtual machines to other servers to facilitate server repairs. In this manner service can continue. In order to test a new system a new partition can be created for testing the system while the original system continues to run uninterrupted in another partition [SN2005]. Thus the new system does have to be installed over the old version. Faults in the new system could otherwise have resulted in downtime.

Flexible Hardware Configuration

In a virtual system a configuration can be different from the actual hardware [Gol1973]. Devices not present in the real system can be simulated. Virtual resources such as memory or CPUs can be added or removed. This is useful for assigning a subset of the available hardware resources to a new system to prevent unrealistic user expectations of performance during the testing phase. Other applications such as facilitation of the development of systems software are beyond the scope of this discussion.

Support for Different Operating Systems

Some partitioning systems support concurrent execution of multiple operating systems or multiple versions of the same operating system. Goldberg included this advantage in a list of advantages provided in a 1973 paper [Gol1973]. This enables applications which require different operating systems or versions of the same operating system to execute in separate partitions. Smith and Nair [SN2005] also discuss the usefulness of this feature.

An example of a system which supports this is VMware [VMW2004a] which supports virtually all of the major x86 operating systems. This is beneficial to data centres which have a wide variety of operating systems. Difficulty managing multiple operating systems was identified as a common data centre problem in Chapter 2. This advantage does not apply to all server partitioning systems as not all support multiple operating systems.

New Servers can be Provisioned More Rapidly

Instead of acquiring a new server, a new virtual server can rapidly be provisioned on an existing server. This new partition provides an execution environment similar to a separate server. This is described by Hewlett Packard [HP2002].

3.6 Applications of Virtualisation in Data Centres

Two types of data centres were identified in Chapter 2, namely Internet data centres and enterprise (corporate) data centres. The most common applications of virtualisation in each of these data centre types are described here:

Internet Data Centres

Shared Hosting

Service providers often make use of shared hosting. A single server is shared by multiple customers [RMS+2000, WSG2002b]. This is also known as virtual web hosting, and is one of the applications of virtualisation listed by Tucker and Comay [TC2004], Barham *et al.* [BDF+2003] and Surányi *et al.* [SHH+2005]. Each customer's data is isolated from other customers sharing the same server. The ease with which virtualisation systems facilitate the implementation of quality of service (QoS) guarantees also makes virtualisation appealing to service providers. This was discussed in the previous section and explored further in Chapter 5.

Enterprise (Corporate) Data Centres

New System Testing and Transition

The isolation qualities provided by virtualisation enable a new application or operating system to be tested alongside a production system in a separate partition. Instead of purchasing a development server a new system can be testing in a development partition without compromising the integrity of the production system. This can reduce or eliminate the need for a separate server used purely for testing. One advantage of this approach is that the test environment is identical to the production system. Hewlett Packard provides information about a number of case studies of organisations which used partitions to facilitate new system testing and transition [HP2002]. Smith and Nair [SN2005] also discuss this application.

Server Consolidation

A traditional definition of server consolidation provided by IBM [IBM2005c] is: *"the process of merging the uses of two or more servers to a single server"*. A single larger server is used to replace multiple smaller servers. This is linked to improved utilisation as discussed in the previous section. The problems of server sprawl and poor resource utilisation were identified as common data centre problems in Chapter 2. A decision may also be taken for a new system to purchase a single large server instead of a number of low-end servers. Each tier of a multitier application can be deployed in a separate partition instead of purchasing separate servers for each tier. Multitier applications were examined in Chapter 2. Hewlett Packard provides information about a case study in which this approach was taken [HP2002]. Server consolidation is a common application of server virtualisation as noted by Barham *et al*. [BDF+2003] and Tucker and Comay [TC2004]. Server consolidation is frequently cited by vendors as an application of virtualisation [HP2002], [SM2005d] and [IBM2005a].

3.7 Conclusion

Virtualisation research has a rich history. The principles on which current virtualisation systems are based originated in the 1960s and 1970s. An interesting observation is that the majority of virtualisation papers were published either in this period or in the last ten years. Despite decades of research, different sources still interpret the terminology differently. Terms such as virtual machine and hypervisor are not used consistently by different researchers. This introduces some confusion when classifying different virtualisation systems. The classification of Smith and Nair [SN2005] formed the basis of the classification discussion in this chapter. This source provides what is almost certainly the only recent classification framework which includes all of the major virtualisation approaches in current use. Given the wide variety of virtualisation systems and approaches formulating this type of classification is a non-trivial problem. This is a fairly high-level classification. In addition to the many approaches to virtualisation there are also many variations in the implementation of each approach. Add to this the wide variety of optimisation options available and changes between different versions of systems.

In this chapter virtualisation was explored from a theoretical perspective, focusing on research sources. The advantages and common applications of virtualisation were identified. This chapter forms part of the theoretical foundation for the framework presented in Chapter 5. In order to obtain a deeper understanding of virtualisation systems, seven current virtualisation systems are examined in Chapter 4. The focus of Chapter 4 will be on information provided in the product manuals of these systems.

Chapter 4: Current Virtualisation Systems

4.1 Introduction

The purpose of the framework which will be presented in Chapter 5 is to evaluate server virtualisation systems in order to compare them. Data centres and the theoretical aspects of virtualisation were explored in the previous two chapters. This chapter is the third of three literature study chapters. In order to construct this framework, a solid understanding of current virtualisation systems is required. To this end seven current virtualisation systems will now be examined. As the focus of the framework is on server virtualisation in the data centre, the systems reviewed here are representative of those which are most likely to be used in a data centre environment. This resulted in virtualisation systems from the major vendors being selected. These vendors include International Business Machines (IBM) [IBM2004b], Sun Microsystems [SM2004d], Hewlett Packard (HP) [HP2004e], VMware [VMW2004a] and Microsoft [MS2004b]. Systems from these vendors are described in this chapter primarily from the point of view of an administrator of these systems. This is because information had to be gathered primarily from server manuals and other system documentation.

This chapter includes the following sections:

- Current Virtualisation Systems
    - Solaris 10 Containers
    - Sun Dynamic System Domains
    - HP Node Partitions (nPars)
    - HP Virtual Partitions (vPars)
    - IBM Logical Partitions (LPAR)
    - VMware ESX Server
    - Microsoft Virtual Server

4.2 Current Virtualisation Systems

Solaris 10 Containers (Container-Based System)

As indicated by the title, Solaris 10 Containers [SM2005d] take a container-based approach to implementing virtualisation. Sun uses the term Solaris Container to refer to Zones which make use of the Solaris Resource Manager. Zones provide security and namespace isolation, while the Resource Manager is used to manage resource allocation.

Zones were available in Solaris 9, but additional support for virtualisation was added for Solaris 10 [SM2005b]. This discussion will focus on Zones in Solaris 10.

The Resource Manager will be examined first, followed by a discussion of Zones. The Solaris Containers Administration Guide [SM2005d] provides additional information about containers.

Resource Manager

The Solaris 10 Resource Manager is used to manage resources such as CPUs and memory. Resource Manager can be used to manage resources of processes assigned to tasks or projects. This discussion will only consider this tool as it applies to assigning resources to zones.

CPU Resource Allocation

The default Solaris scheduler is the timesharing scheduler. With this scheduler all zones have equal access to processor resources. In order to provide zones with guaranteed CPU resources, the fair share scheduler (FSS) can be used. Each zone is assigned a number of CPU shares. These shares are used by the scheduler to determine the quantity of CPU resources to allocate to each zone. The actual number of shares assigned to a zone is not enough to determine the CPU allocation it will receive. The CPU allocation a zone receives depends on the number of shares assigned to it relative to the total number of shares assigned to other zones. The actual proportion of CPU resources allocated to each zone is derived using the formula which is depicted in Figure 4.1.

$$\text{allocation}_{\text{zone}}\,i = \frac{\text{shares}_{\text{zone}}\,i}{\sum_{j=1..n} (\text{shares}_{\text{zone}}\,j)}$$

(where only booted zones are considered)

Figure 4.1: Zone CPU Resource Allocation (adapted from figure in [SM2005d]*)*

If a zone is not using all of the CPU resources assigned to it, those resources can be assigned to processes belonging to other zones which need them. Thus it is possible for a zone to receive more resources than it was allocated shares for. This ensures that CPU cycles are not wasted. CPU shares only limit resource usage when there is contention for resources between zones.

Even if a zone has been assigned many shares, actual CPU utilisation depends on the degree of parallelism of the workload in each zone. A zone without parallel processes or threads actively using the CPU cannot take full advantage of a multi-CPU system. Remaining CPUs can be used by other zones. An example of this is outlined by Sun [SM2005d]. This example was adapted, and appears in Figure 4.2. Zone B is only able to use one CPU because it only has one process (single-threaded), despite having a much larger shares allocation than Zone A.

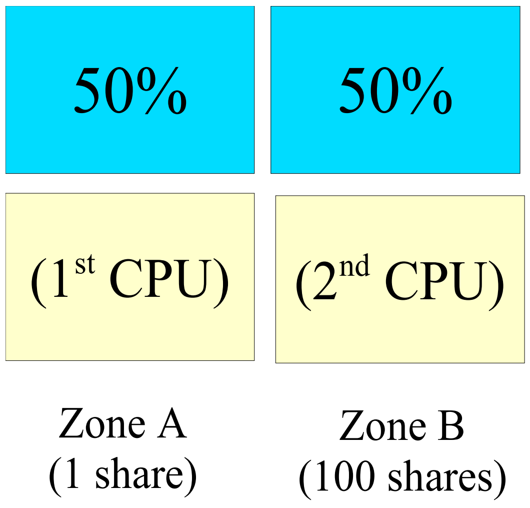


Figure 4.2: Zones Shares Allocations (adapted from figure in [SM2005d]*)*

CPUs can also be grouped into processor sets. Processor sets can be used to dedicate a subset of the CPUs to one or more zones. Since processor sets isolate physical CPUs, processor utilisation will typically be lower than when using only scheduler-based isolation. Even if excess capacity is available in one processor set, CPU resources will not reassigned by default. Dynamic resource pools, which will be covered later in this discussion, enable CPUs to be reallocated from one zone to another. A zone can only be assigned to one processor set. The fair share scheduler can be used to allocate CPU resources within a processor set. Resources within a processor set are assigned to zones based on the number of shares assigned to each booted zone assigned to that processor set.

Memory Allocation Controls

Physical memory usage of zones can be limited. This is enforced by the resource capping daemon. This process periodically measures the memory usage of all processes within a capped zone. The default sampling interval of this daemon is five seconds. The minimum is one second. If memory usage is found to exceed the cap value when sampled, physical memory usage is reduced by swapping to disk. Memory usage is not monitored between samples. As a result, memory usage may briefly exceed the cap value specified. Shortening the sampling interval reduces the chances of memory usage significantly exceeding the cap value. A disadvantage of using a shorter sampling interval is that it increases overhead. This is especially true if a large number of processes need to be monitored. If the memory cap for a zone is set too low, it will result in increased swapping to disk, which negatively affects system performance. Thus it is important to choose a suitable memory cap.

Memory caps are only enforced once physical memory usage of the system climbs above a certain threshold value. This threshold value can be set. It is therefore possible for the physical memory usage of a zone to exceed its memory cap. This occurs if the system memory capping enforcement threshold has not been exceeded. This enables capped zones to use more memory when the system has memory to spare. The overhead of the resource capping daemon is also reduced if the threshold has not been exceeded.

Dynamic Resource Pools (DRPs)

Dynamic resource pools can be used to dynamically reallocate resources which are assigned to zones. Currently this functionality is restricted to managing CPUs allocated to zones. Resources are allocated based on objectives. There are three types of objectives which can be set, namely locality, wt-load and utilisation.

A locality objective for a pool can be set to "tight", "loose", or "none" [SM2005d]. Locality of components such as CPU and memory are measured using latency to determine relative distances between components. If locality is set to "tight" the system will attempt to use configurations which maximize locality. Similarly, if locality is set to "loose", the system will attempt to use configurations which minimize locality. A locality value of "none" results in locality not being taken into account when creating a configuration. This is the default setting.

A utilisation objective can be set to achieve specific resource utilisation goals. The utilisation objective can be set to "less than", "greater than" or "about" a specific value [SM2005d]. A range using both "less than" and "greater than" can be specified.

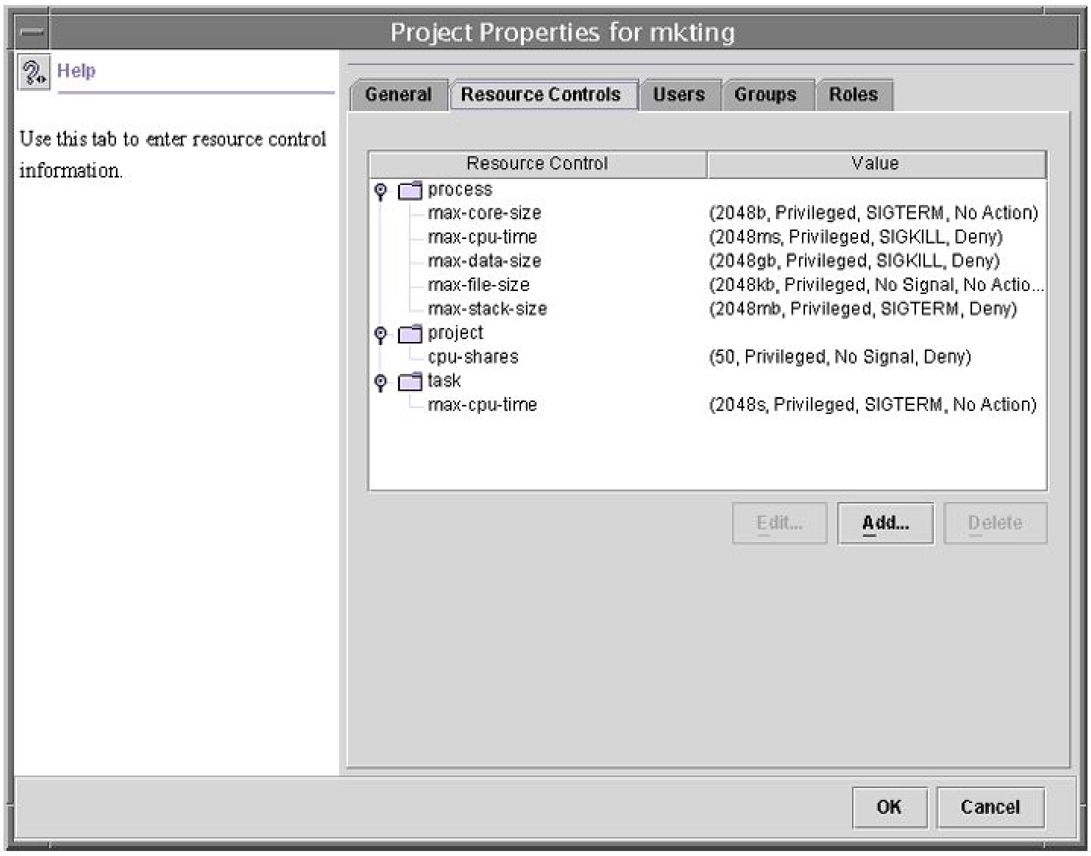
If a wt-load objective is set the system will try to match resource supply to demand.

A default pool and a default processor set always exist when the pools feature is enabled. Resource pools have a processor set (pset) property. A pset can have pset.min and pset.max properties set. These values specify the minimum and maximum number of CPUs which can be assigned to a processor set. These settings are used as constraints when resources are being assigned to pools. Each resource pool can use a different scheduling algorithm.

A processor can be dedicated to a particular pset using the pset.pinned property. This prevents that processor from being assigned to another partition.

The pools dynamic resource controller (poold) is responsible for managing pool resources. It monitors information about the system to determine how to meet the objectives. If an objective is not being met corrective measures, such as moving a CPU from one processor set to another, can be taken. A decision history stores a record of measures taken. This is to prevent the system from repeating a reconfiguration which did not yield a successful outcome.

The resource management features of Solaris 10 are all available using command line utilities. A subset of these can be accessed using the Solaris Management Console (SMC) GUI. A screenshot of the SMC GUI appears in Figure 4.3.



Use this tab to enter resource control information.

Resource Control	Value
process	
max-core-size	(2048b, Privileged, SIGTERM, No Action)
max-cpu-time	(2048ms, Privileged, SIGKILL, Deny)
max-data-size	(2048gb, Privileged, SIGKILL, Deny)
max-file-size	(2048kb, Privileged, No Signal, No Actio...
max-stack-size	(2048mb, Privileged, SIGTERM, Deny)
project	
cpu-shares	(50, Privileged, No Signal, Deny)
task	
max-cpu-time	(2048s, Privileged, SIGTERM, No Action)

Figure 4.3: Solaris Management Console [SM2005d]

Solaris Zones

Using Solaris Zones a single instance of Solaris 10 can be partitioned into multiple isolated "execution environments" [SM2005d]. Each of these zones shares a common operating system instance with a single OS kernel. Zones provide the appearance of multiple operating system instances, when there is in fact only a single instance of Solaris. For more information about the container-based approach to server virtualisation refer to Chapter 3.

Zones enable multiple virtual servers to be created from a common physical server or system domain. Zones which are used in conjunction with resource management controls are referred to as Solaris Containers.

There are two types of zones, global and non-global. The global zone is the only zone which can observe and manage non-global zones. Zones can only be created and destroyed from this global zone. It is also the only zone which has access to all file systems and automatically has access to all devices. The global zone is the global operating system instance shared between all zones.

Zones can be configured using command lines tools, or a web-based graphical interface [SM2005a]. A sample script is provided in the Solaris Containers Administration Guide [SM2005d], which can be customized to automate zone creation.

As can be seen in Figure 4.4, the file system of a non-global zone is a subtree of the file system of the global zone. The root directories of the non-global zones a,b and c can be seen in this diagram. The dotted lines indicate loopback mounts which are used to provide a mapping between non-global zones and directories in the global zone. The root directory of a non-global zone is referred to as its zonepath.

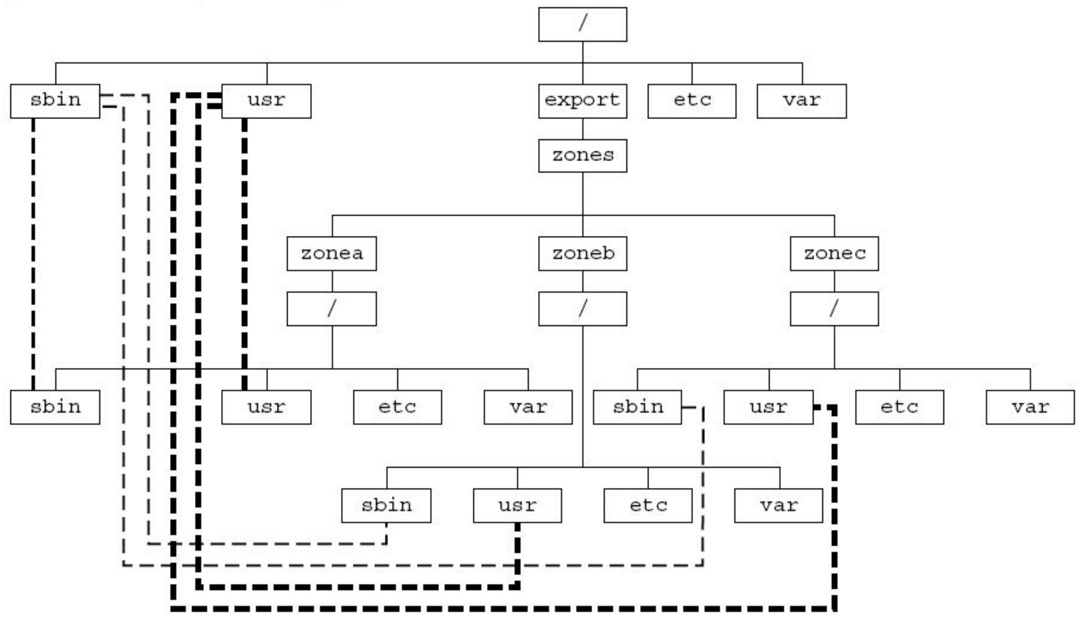


Figure 4.4: Filesystem Subtrees for Non-global Zones [SM2005d]

Each new non-global zone is claimed to require about 100 MB of disk space for the standard Solaris packages. Each zone has its own set of software packages installed. Disk space limits can be placed on zones using partitions. This prevents one zone from consuming a disproportionate amount of disk space.

Up to 8192 zones can exist simultaneously on a single system, although it is more likely that the actual number will be limited by the physical resources available. Non-global zones can be booted separately. These reboots are much faster than system reboots.

Each zone, whether global or non-global is assigned an ID. The system assigns the ID 0 to the global zone. Non-global zones are completely isolated from each other in software. Processes in different zones cannot signal each other. Monitoring of processes in other zones is prohibited. Each zone can have one or more IP addresses. Zones sharing a network card using multiple logical interfaces cannot read each other's traffic despite using the same card. Different zones can also bind to the same network port with different IP addresses simultaneously.

This level of isolation offers a number of advantages. If the security of one non-global zone is compromised, the security breach is restricted to that zone. Different versions of the same application can exist simultaneously in different zones without conflict. Figure 4.5 depicts different versions of various applications consolidated onto the same server. The distinction between the global zone and the non-global zones is clear.

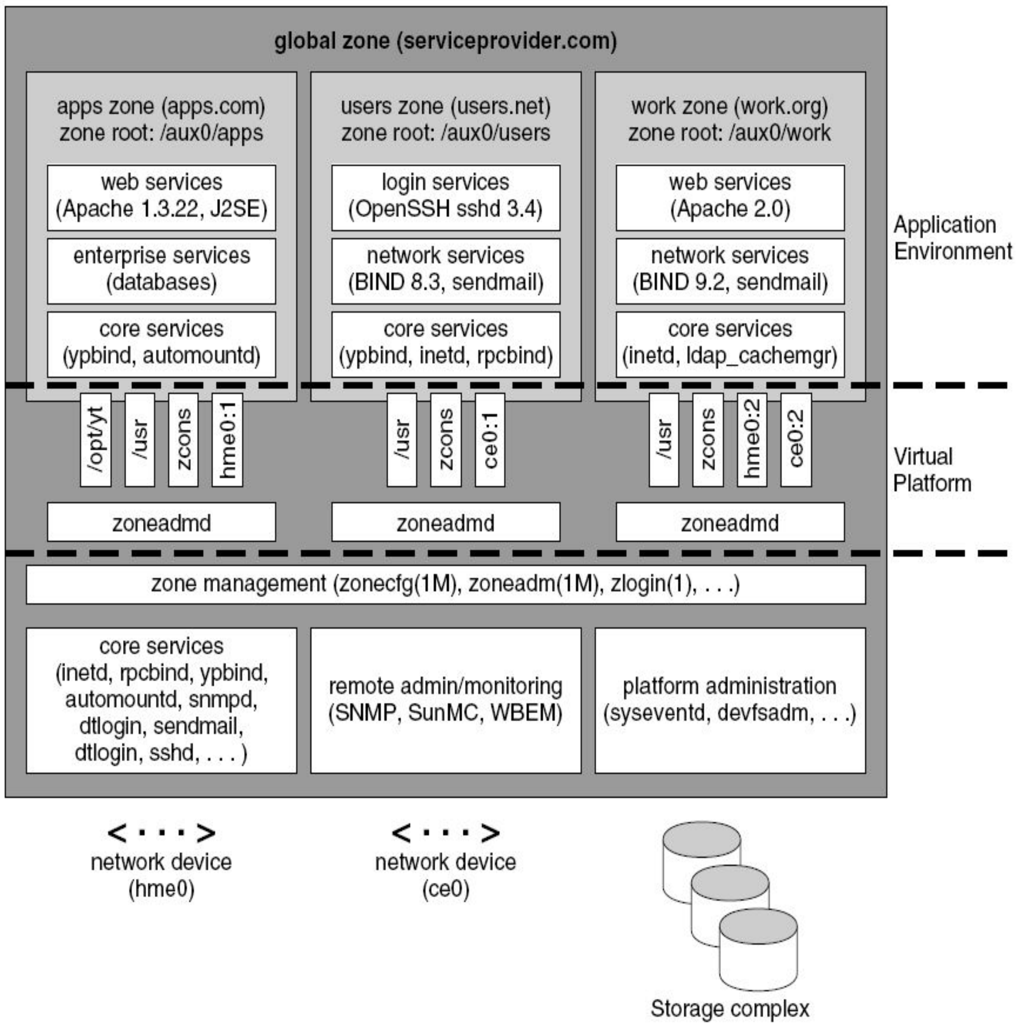


Figure 4.5: A Server Hosting Various Applications within Zones [SM2005d]

The performance overhead of zones is claimed to be very low. This is illustrated in Figure 4.6, which presents the performance measurements of Tucker and Comay [TC2004].

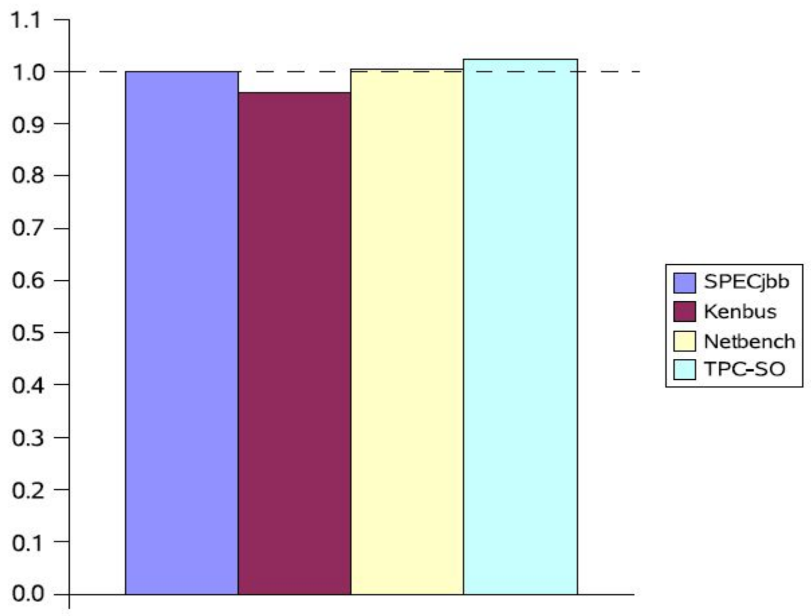


Figure 4.6: Zone Performance Relative to Non-Zone Performance

Sun Dynamic System Domains (Physical Partitioning System)

Sun Dynamic System Domains [SM2004f] provide physical partitioning. It enables multiple instances of Solaris to be installed on a single physical server. A single Sun Fire 15K for example can have up to 18 domains [SM2003b]. Domains are isolated from software and hardware faults in other domains. This is similar to Hewlett Packard's nPartitions [HP2004i].

Each domain has a separate boot disk with an instance of Solaris installed [SM2003b]. Resources assigned to domains are located on system boards. Currently CPU/Memory boards can contain up to 4 CPUs and 32 GB of RAM. An I/O board contains 4 PCI slots. Resources are assigned to domains along board boundaries. Each domain can be assigned a number of system and I/O boards. Figure 4.7 depicts a system board.

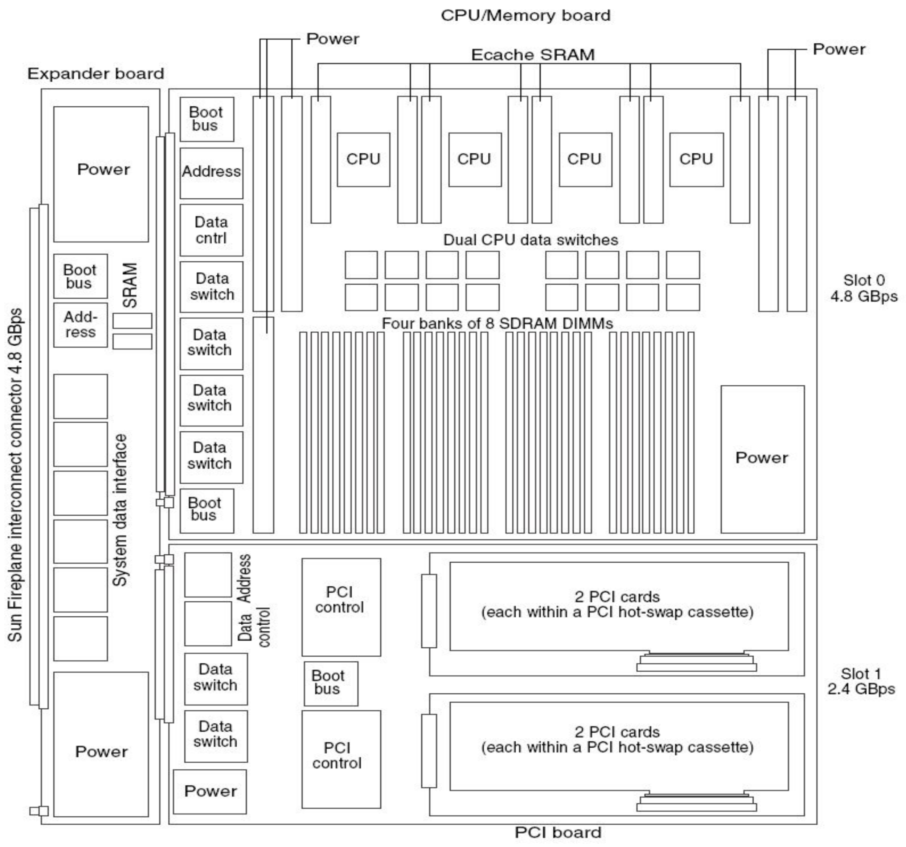


Figure 4.7: A System Board [SM2003b]

Dynamic Reconfiguration (DR) is used to add, remove or reconfigure components in domains without shutting down the system [SM2004f]. Resources such as memory and CPUs can be dynamically moved between domains [SM2003b]. Figure 4.8 and 4.9 show resources being reassigned in a dynamic reconfiguration operation. A system board (SB1) is being moved from domain B to domain A [SM2001]. No I/O boards (IBs) are reassigned.

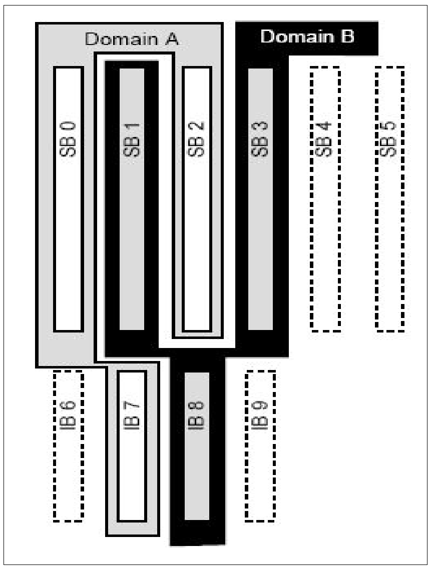


Figure 4.8: Configuration before DR Operation [SM2001]

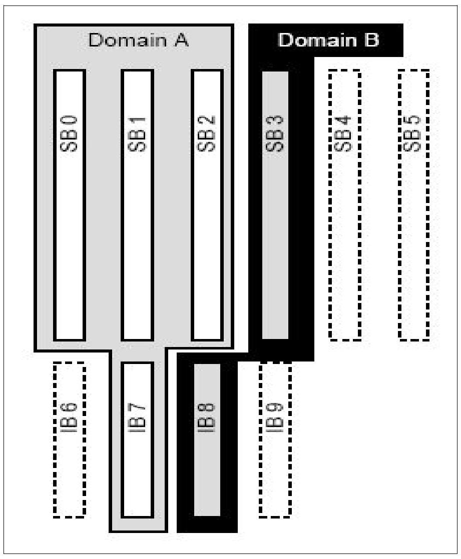


Figure 4.9: Configuration after DR Operation [SM2001]

The advantages and uses of domains are similar to those of other partitioning technologies discussed in this chapter. One of the primary advantages is having the ability to consolidate workloads running on multiple separate servers onto a single server with both hardware and software isolation.

HP Node Partitions (nPars) (Physical Partitioning System)

Node partitions, like Sun Dynamic System Domains [SM2004f], are an example of a physical partitioning system.

Node partitions can be used to partition servers such as the HP Superdome which consists of a number of cell boards (or cells). Each of these cells contains CPU and memory resources [HP2004b].

A single physical server can be partitioned into multiple node partitions (nPars) [HP2004i] each consisting of one or more cells [HP2002]. This effectively divides the server into multiple smaller servers along cell boundaries. A 64-way HP Superdome for example can be divided into a maximum of 16 nPartitions (one cell per nPar). More than one cell can be assigned to a node partition.

Hard partitioning provides electrical isolation between nPars. This prevents component failures in one partition from affecting others [HP2002]. Figure 4.10 depicts the architecture of a server containing 2 cells. Each cell can be assigned to a separate Node Partition. Note the physical separation of the two boards.

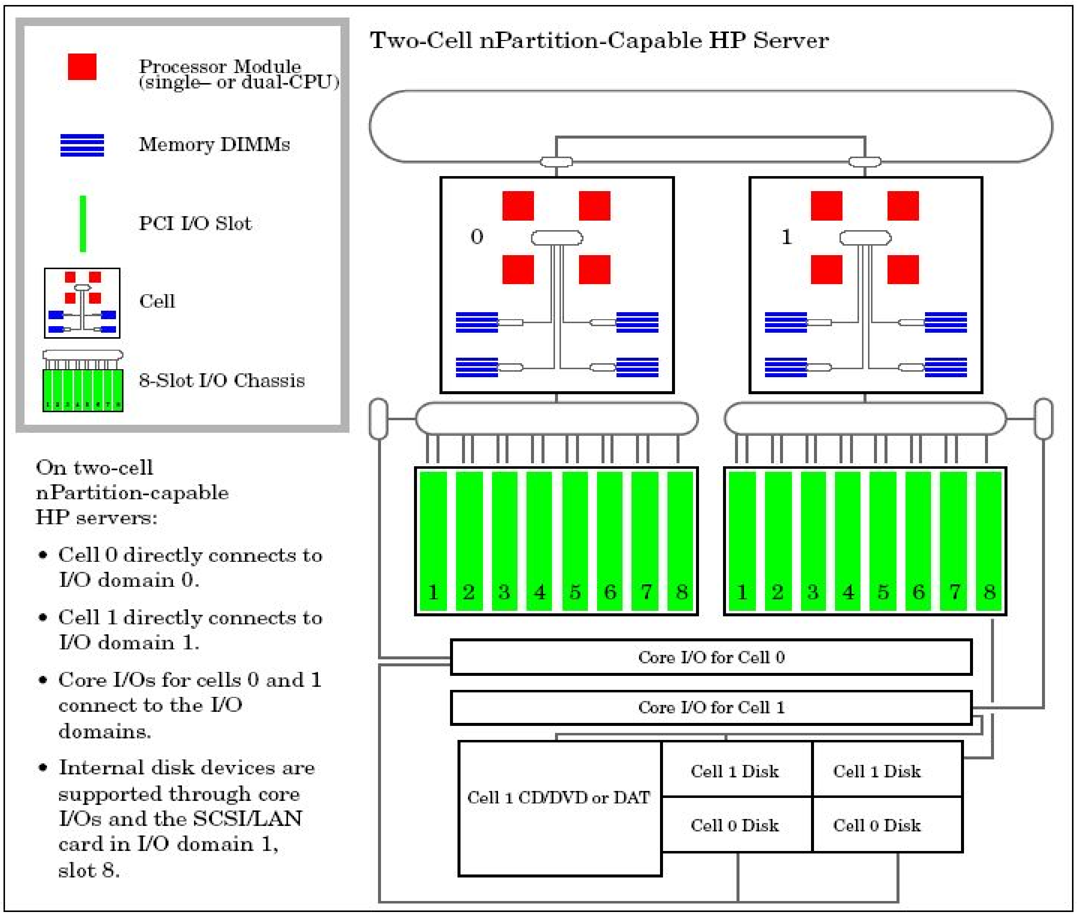

Processor Module (single– or dual-CPU)
Memory DIMMs
PCI I/O Slot
Cell
8-Slot I/O Chassis
Two-Cell nPartition-Capable HP Server
0
1
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
Core I/O for Cell 0
Core I/O for Cell 1
Cell 1 CD/DVD or DAT
Cell 1 Disk
Cell 1 Disk
Cell 0 Disk
Cell 0 Disk
On two-cell nPartition-capable HP servers:
• Cell 0 directly connects to I/O domain 0.
• Cell 1 directly connects to I/O domain 1.
• Core I/Os for cells 0 and 1 connect to the I/O domains.
• Internal disk devices are supported through core I/Os and the SCSI/LAN card in I/O domain 1, slot 8.

Figure 4.10: A Hewlett Packard Server with Two Cell Boards [HP2004c]

Each nPartition has a separate instance of the operating system [HP2004b]. Thus nPars provide both hardware and software isolation. Operating system (OS) errors in one partition will not affect others One advantage of this approach is that each partition can use a different operating system version. Node partitions on PA-RISC systems support HP-UX. Different operating systems can be used in different nPars with Itanium-based Integrity servers [HP2005e]. These servers support Windows, HP-UX, OpenVMS and Linux for nPartitions [HP2005i]. On PA-RISC systems an nPartition can contain multiple vPars (discussed in the next section) which each run a separate operating system instance.

Hardware isolation also ensures that each nPartition has dedicated CPU, memory and I/O resources assigned to it [HP2004b]. Partition Manager is used to manage nPartitions. If resources need to be added to or removed from a nPar, that nPar has to be rebooted. To prevent disruptions to other nPartitions, each nPar can be rebooted independently.

HP Virtual Partitions (vPars) (Logical Partitioning System)

Virtual Partitions (vPars) enable multiple instances of the HP-UX [HP2005c] operating system to coexist within a single node partition or server [HP2004b]. Node partitions can be further subdivided into multiple vPars [HP2004j]. Figure 4.11 depicts the relationship between nPars and vPars.

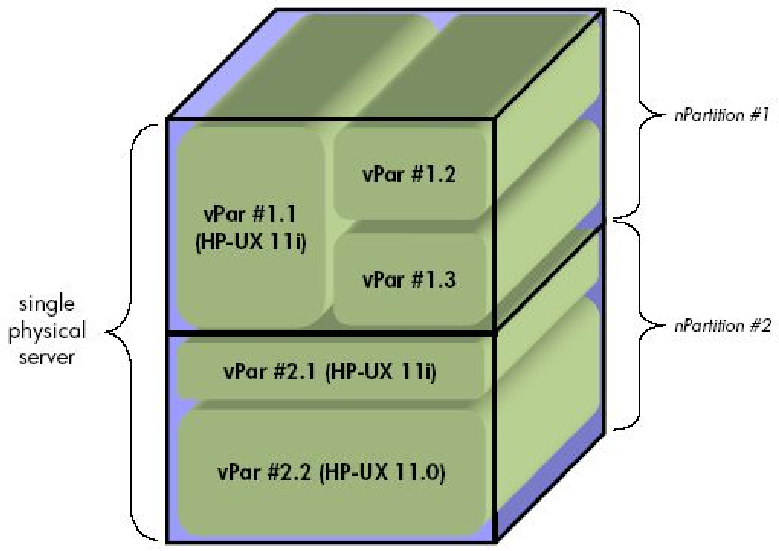


Figure 4.11: Virtual Partitions [HP2004b]

Isolation

Virtual partitioning is an example of software partitioning. Unlike nPars which provide physical partitioning, vPars are not isolated from hardware failures in other partitions. Virtual partitions are isolated from software faults in other vPars such as operating system (OS) panics [HP2003a]. Unlike container-based systems each vPar has a separate instance of the operating system. Each vPar within a server or nPar must run the same version of HP-UX [HP2005h]. Each HP-UX instance may have a different patch level however [HP2004k]. This is very useful if different applications on a server may require different OS patch levels. Figure 4.12 depicts a number of virtual partitions with various patch levels and applications.

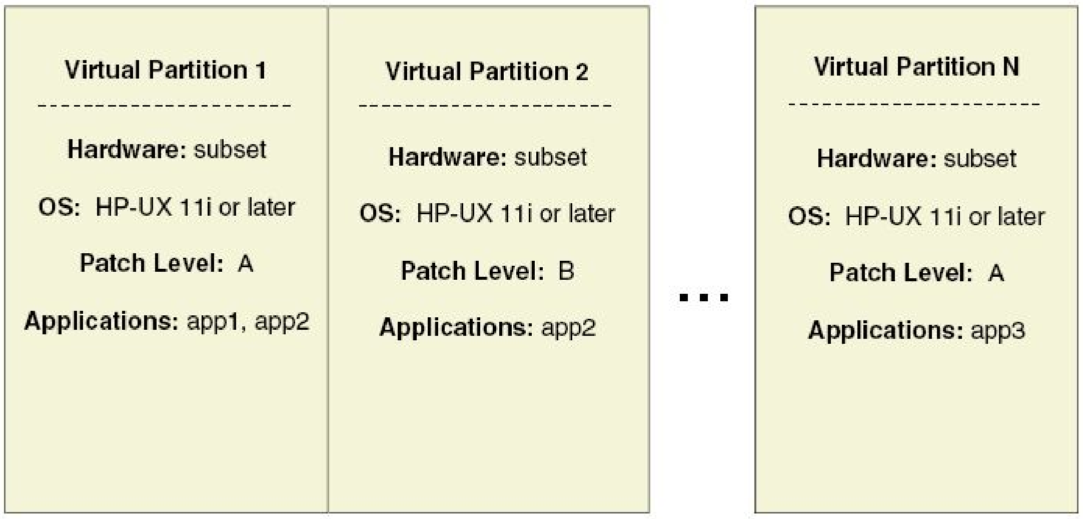


Figure 4.12: vPar Software Isolation and Compatibility [HP2004k]

Virtual Partitions are supported on PA-RISC and Itanium systems, but HP-UX is the only operating system supported..

As a logical partitioning system virtual partitions provide isolation from application faults and OS kernel panics in other vPars. A footnote in [HP2004k] states that this isolation does not extend to panics in the underlying vPar monitor itself.

Only root users can create, modify or delete vPars [HP2003a]. One security concern with vPars is that a root user in one partition can affect other partitions using vPar commands [HP2004k]. These commands could be used to reboot another partition for example.

Resource Management

Each vPar is assigned one or more CPUs. The number of CPUs assigned must be an integer as there is no time-slicing of CPUs between vPars [HP2003a]. Thus the lowest level of granularity provided is a single CPU [HP2004b]. This is better than the multi-CPU granularity offered by nPartitions. Physical memory is allocated to vPars in multiples of 64 MB [HP2003a]. Resources can only be dedicated to one vPar at a time [HP2004j]. This prevents resource conflicts between partitions [HP2004k]. Processors are classified as either bound or unbound. A bound processor can receive

I/O interrupts for a vPar while an unbound CPU cannot [HP2004k]. Each vPar requires at least one bound CPU so that I/O can be processed. Only unbound CPUs can be shifted from one partition to another dynamically (without the need to reboot). It is recommended that more bound CPUs should be assigned to I/O intensive applications. Reallocating memory between partitions requires a reboot [HP2003a]. Figure 4.13 depicts server resource subsets allocated to each vPar.

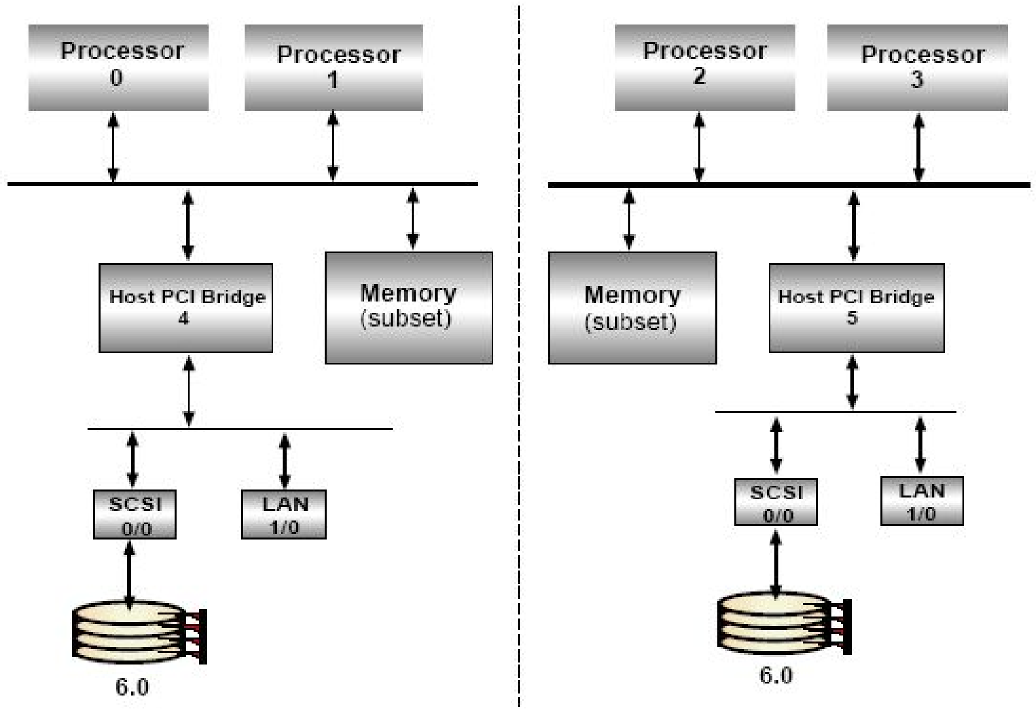


Figure 4.13: Assigning Resources to Virtual Partitions [HP2004k]

Resource requirements for vPars are listed in [HP2004j]. Each partition requires a minimum of 1 CPU, a unique LAN card, a unique boot device and depending on the server between 256 MB and 512 MB of memory for a HP-UX instance. Although at least 1 GB of memory per CPU is recommended.

A virtual partition monitor is responsible for booting vPars [HP2004k]. To create the impression of multiple physical servers the vPar monitor emulates some firmware calls. The vPar monitor also assigns resources to partitions based on settings in the vPars partition database. This database stores information about the settings of vPars.

Figure 4.14 depicts a hard partition which is divided into two vPars. The vPar monitor fits in between the HP-UX instances and the hardware.

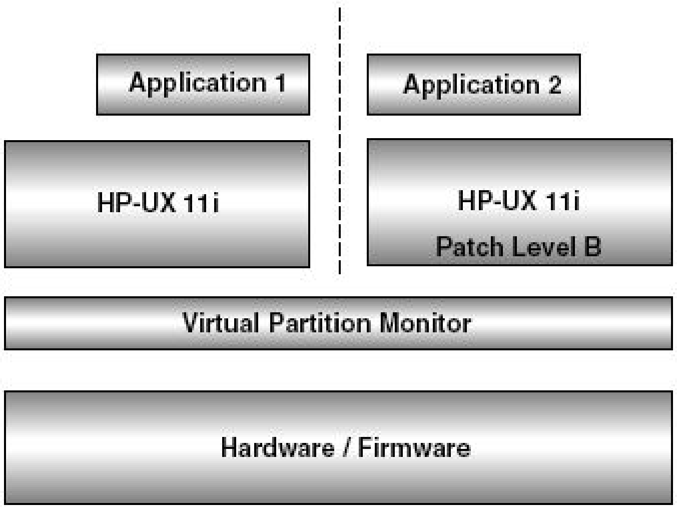


Figure 4.14: Virtual Partitions [HP2004k]

Managing Virtual Partitions with HP-UX Workload Manager

HP-UX Workload Manager (WLM) is covered in this section as an example of a tool for managing resources assigned to partitions with the goal of meeting service level objectives (SLOs). Workload Manager can manage resources within and between separate vPars [HP2004h].

Goal-based SLOs specify measurable goals. These goals can either be metric or usage goals. Usage goals are used to ensure CPU utilisation remains at a certain level. If usage drops below a certain threshold, CPUs can be assigned to other groups. Metric goals specify other types of measurable goals such as transactions per second [HP2004d].

Priorities can be assigned to a service level objective. A SLO priority of 1 is the highest permitted. Priorities of 2 or more are of lesser importance. Lower priority SLOs are only considered if all higher priority SLOs are currently being met.

Constraints can be used to ensure that CPU resources assigned to a certain group remain within a specified range.

The conditions in an SLO specify criteria that must be met before a SLO is activated. An example could be a certain number of users need to connect to a system before the SLO is in effect.

Sometimes additional metrics are required to effectively measure the performance of applications. For this reason Workload Manager toolkits are available to gather metrics for popular applications such as Oracle and Apache [HP2004g]. These metrics can be used as conditions and goals of SLOs.

Workload Manager can move processors from one vPar to another vPar dynamically in order to meet service level objectives.

IBM Logical Partitions (LPARs) (Logical Partitioning System)

The term LPAR is used to refer to partitioning of both POWER-based servers and IBM mainframes such as the zSeries [IBM2004d]. In this chapter only the POWER-based pSeries systems [IBM2004e] will be considered.

Multiple operating system instances can be hosted simultaneously on the same system using LPARs [BE2004]. AIX and Linux are supported on pSeries servers.

The Hypervisor

The POWER hypervisor forms a layer between the hardware and the guest operating systems. This acts as an abstraction layer [IBM2004a] and is implemented in firmware [IBM2003a].

As the name suggests IBM Logical Partitioning is a logical partitioning system. This approach to virtualisation was discussed in Chapter 3.

The POWER hypervisor makes use of paravirtualisation [BE2004]. The operating system kernel is modified to make use of special hypervisor calls. These calls improve the efficiency of LPARs by enabling partitions to release processor resources when idle. Support for the hypervisor is build into the POWER4 and POWER5 processors. This is implemented by providing a higher privilege state called the hypervisor state. This state restricts access to certain resources to the hypervisor alone.

A Hardware Management Console (HMC) is used to manage partitions. This is a Linux based appliance which connects to the system via a serial connection [IBM2003b]. A screenshot of the HMC user interface can be seen in Figure 4.15.

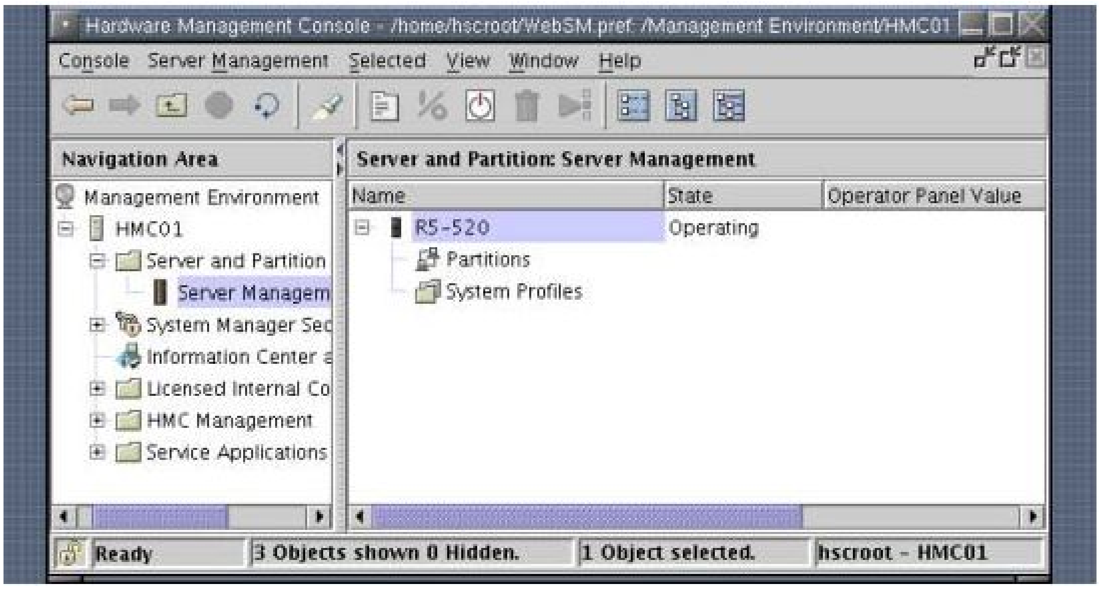


Figure 4.15: Hardware Management Console [IBM2004h]

Partition Types

There are two types of partitions supported, namely, dedicated and shared. Each of these partition types will now be discussed.

Dedicated Partitions

With dedicated partitions resources can only be allocated to one partition at a time. There is no sharing of resources between partitions [IBM2003a]. Systems based on the POWER4 processor only support dedicated partitions. Both memory and CPUs are dedicated to the partition. Figure 4.16 depicts the assignment of dedicated processors to each partition. This figure is of an iSeries system, but the principle is the same for pSeries servers.

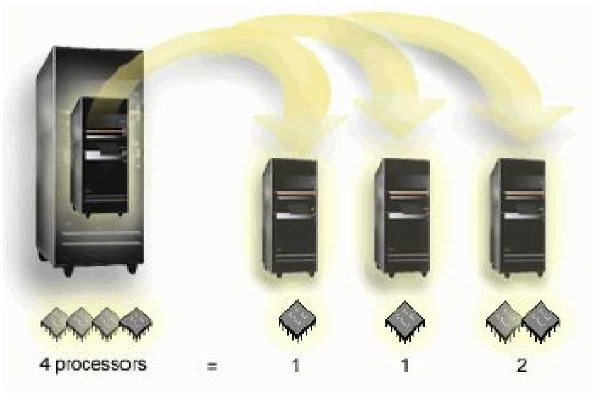


Figure 4.16: Dedicated Processor Partitions [IBM2002b]

Shared Partitions

The second type of partition is a shared partition. This feature is also referred to as Micro-Partitioning [IBM2004f]. Shared partitions enable processors to be shared between more than one partition [IBM2004a]. All shared processor partitions share a common pool of physical processors. Each partition can be assigned a minimum of 1/10th of a CPU [BE2004]. This is implemented using virtual processors, each of which represent between 10% and 100% of the capacity of a physical processor [IBM2004f]. This feature is implemented in the underlying hardware and the hypervisor. Each virtual processor is dispatched to a physical CPU at regular intervals by the hypervisor. The hypervisor can dispatch each virtual CPU to any of the physical CPUs available [IBM2005b]. It attempts to maximise memory affinity when doing so. The dispatching process takes the form of a dispatch wheel. This is illustrated in Figure 4.17.

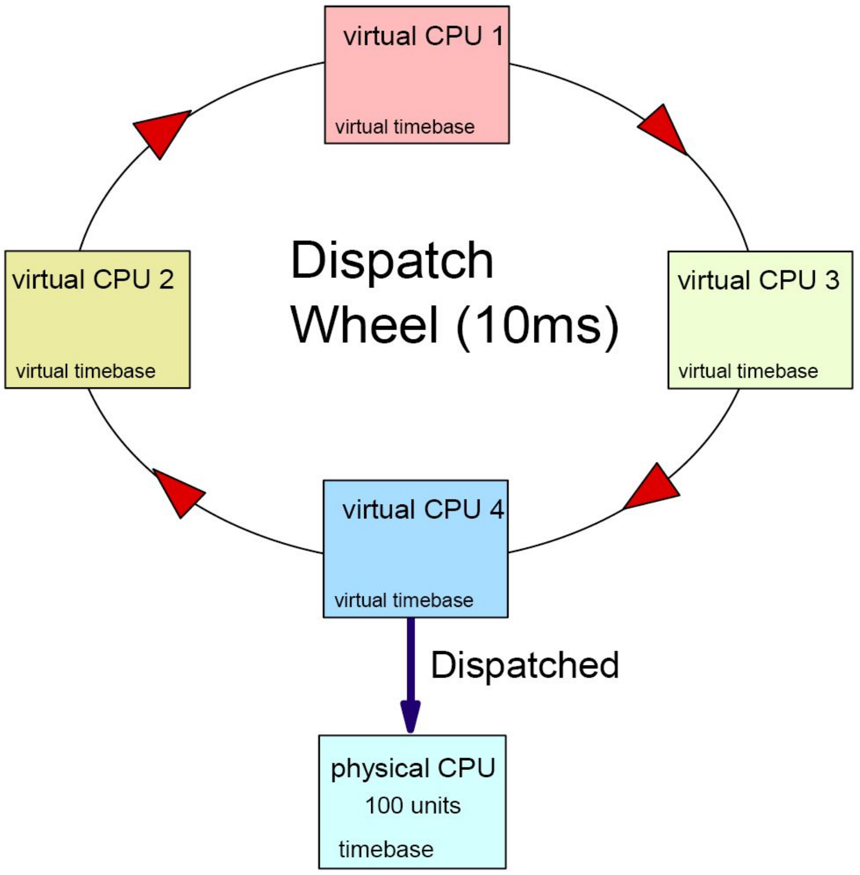


Figure 4.17 The Dispatch Wheel [IBM2004f]

The duration of each rotation of the dispatch wheel is 10 milliseconds. There is a latency between successive timeslices on a physical CPU as a result. The maximum latency which can occur is 18 milliseconds. This is occurs if a virtual CPU representing the minimum entitlement of 10% runs in the first millisecond of one dispatch cycle and in the last millisecond of the next.

Shared partitions are only supported since the POWER5 processor [IBM2004f]. This feature extends the dedicated partitioning functionality available on POWER4-based systems.

Having a larger number of virtual processors per physical processor results in reduced efficiency due to each virtual processor getting smaller time slices on a physical processor. It also results in decreased cache efficiency due to frequent memory context switches.

Mixing Partition Types

Both dedicated and shared processor partitions can be used simultaneously on the same system [IBM2004f]. In a dedicated processor partition physical processors are assigned to that partition and cannot be shared with other partitions. This is the only type of partition available on POWER4 based systems. As discussed previously, shared processor partitions make use of virtual processors to share physical processors between partitions. In Figure 4.18 dedicated and shared processor partitions are being used on the same system. Eight CPUs are shared in a pool between six partitions. Each partition is assigned a number of virtual CPUs from this pool of processors. Three other dedicated partitions are also depicted.

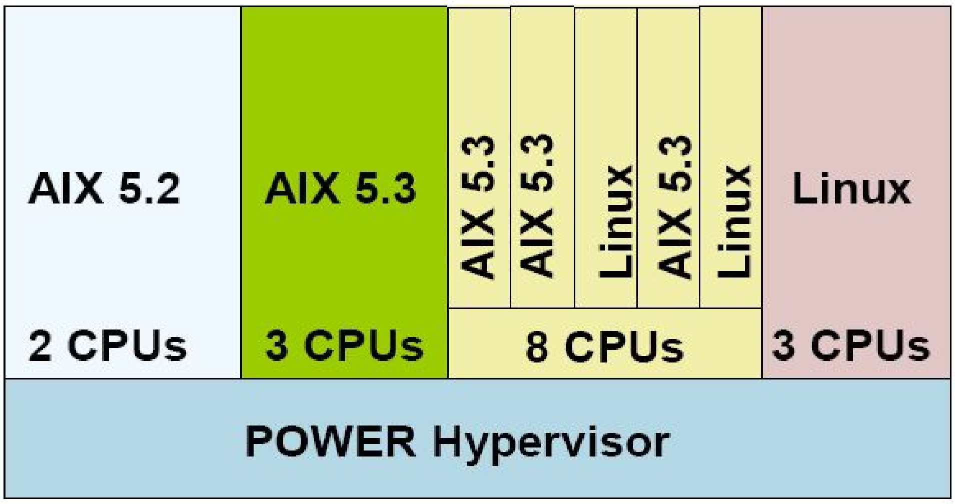


Figure 4.18: Mixing Partition Types [IBM2004f]

Virtual I/O

Multiple virtual I/O devices can share a single physical device [IBM2004f]. This functionality is provided by the Virtual I/O Server which is a special partition dedicated to providing I/O support to other partitions [IBM2005b]. It is interesting to note that the use of specialised virtual machines providing services to neighbouring virtual machines dates back to the 1970s [BFH+1975]. Virtual I/O devices can be assigned to client partitions which share physical devices in the Virtual I/O Server partition [IBM2005b]. This enables a system to have more partitions than it has slots for I/O devices [IBM2004f]. Partitions can have a mixture of physical devices dedicated to them and virtual I/O devices. Examples of virtual I/O include Shared Ethernet Adapter (SEA) and virtual SCSI [IBM2005b]. Virtual SCSI is shown in Figure 4.19. Client partitions 1 and 2 access a shared disk via the virtual I/O server partition. These client partitions have virtual SCSI client software installed which is used to communicate with the virtual SCSI server software installed in the I/O server partition.

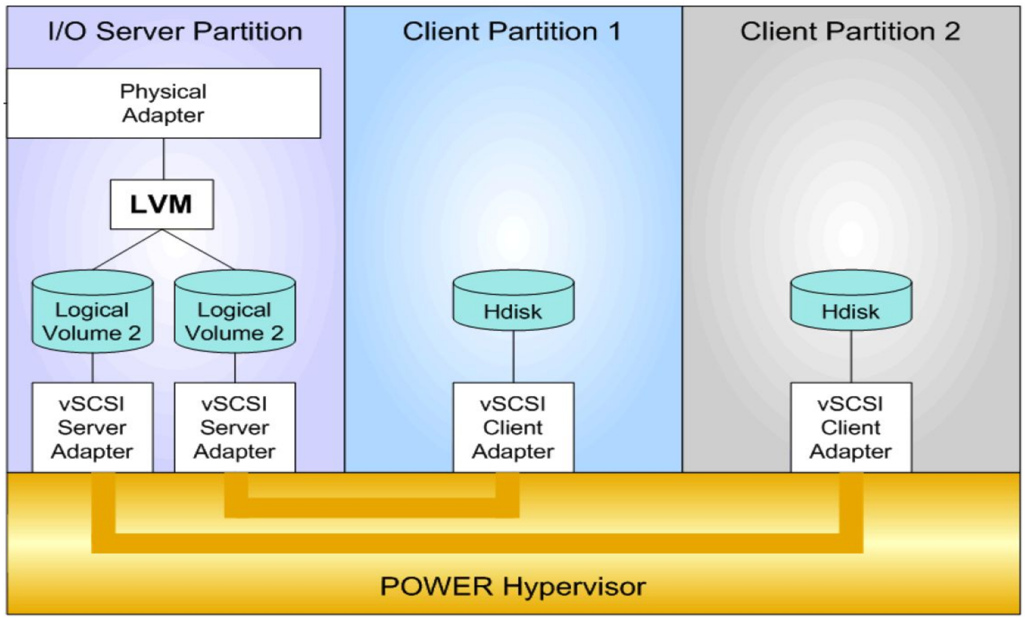


Figure 4.19: Virtual SCSI [IBM2005b]

Similarly, Shared Ethernet Adapter enables a single Ethernet adapter dedicated to the Virtual I/O Server to be used to connect other partitions to the external LAN. Virtual Ethernet provides network connectivity between partitions without having any physical adapters [IBM2004f]. This is implemented using a memory based inter-partition LAN. Figure 4.20 depicts two partitions connected via Virtual Ethernet.

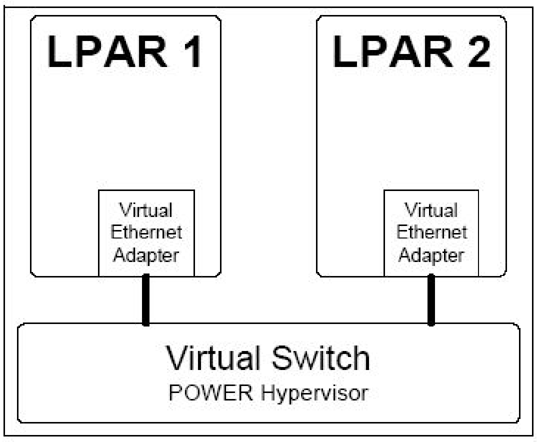


Figure 4.20: Virtual Ethernet [IBM2004f]

Resource Management

Capping Shared Partitions

There are two types of shared processor partitions namely capped and uncapped [IBM2004f]. Capped partitions have a limit on the quantity of CPU resources which they can consume. This is enforced even if there are idle resources available. Uncapped partitions can take advantage of excess capacity in the shared processor pool enabling it to exceed the entitlement for that partition. In Figure 4.21 a capped partition is given a resource ceiling of 9.5 physical processors from the pool of 16 processors. The excess capacity is not used even though it is available.

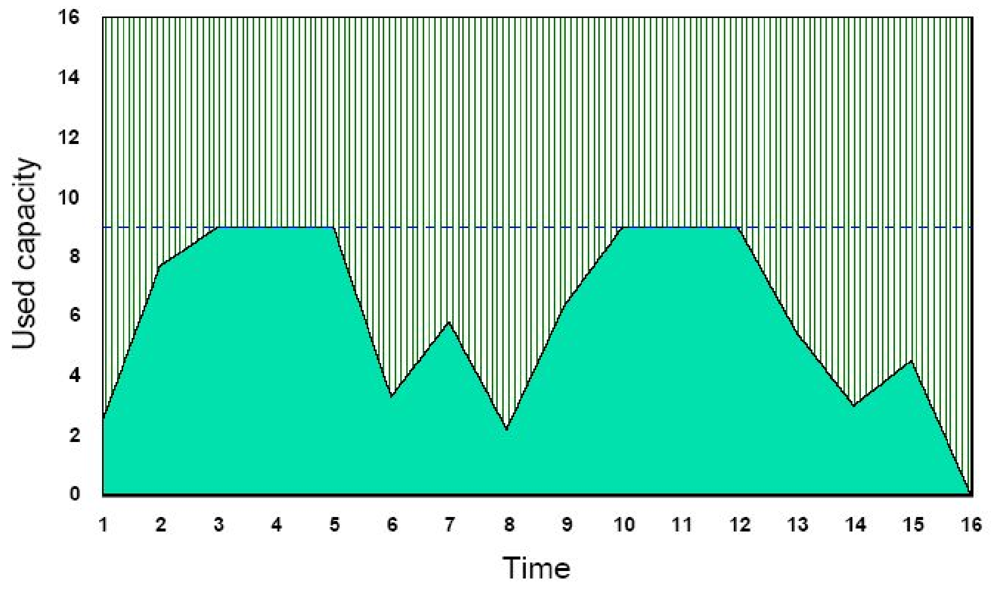


Figure 4.21: A Capped Partition [IBM2004f]

In Figure 4.22 an uncapped partition is permitted to use all available CPU resources in the shared processor pool. It is clear from this Figure that utilisation is higher.

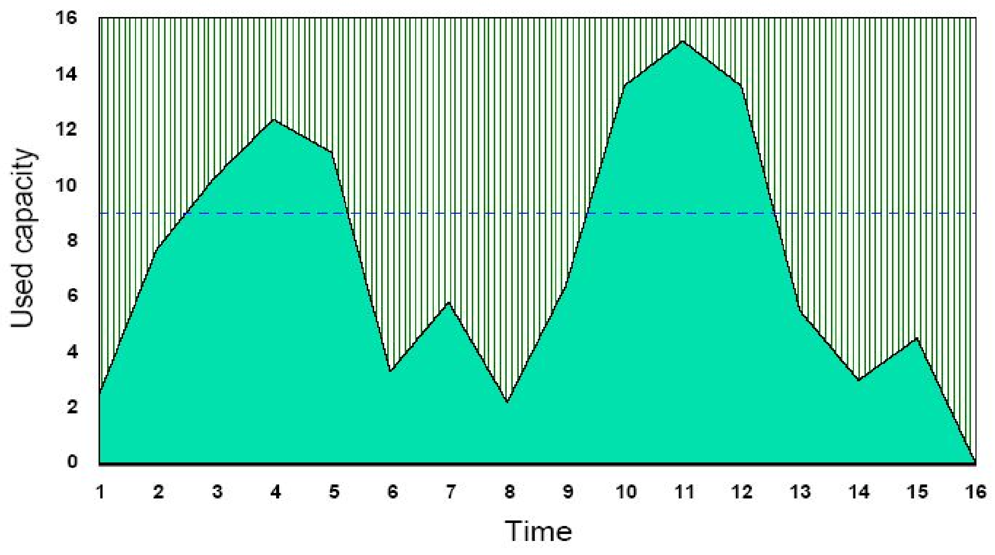


Figure 4.22: An Uncapped Partition [IBM2004f]

Hypervisor Resource Management

Each shared partition is assigned a number of values. The minimum and maximum processing units specify limits on the number of processing units which can be assigned to a partition [IBM2004a]. Each of these is expressed as fractional number of CPUs (e.g. 0.3 or 2.5). Similarly a desired number of processing units which is between the minimum and maximum is also specified for each partition. No partition can have less than 0.1 processing units (10% of a CPU) [IBM2004a]. As mentioned previously, shared partitions can be configured as either capped or uncapped. Uncapped partitions are assigned a weight between 0 and 255. This weight is used to calculate the quantity of excess capacity each uncapped partition is entitled to [IBM2004a]. Excess capacity is assigned to partitions which are competing for resources in proportion to their weightings.

A minimum, maximum and desired number of virtual processors is also specified. The number of virtual processors assigned to a partition will affect the degree of concurrency that can be achieved.

Dedicated partitions have dedicated physical CPUs assigned to them. A minimum and maximum number of processors is also specified. The role of the minimum and maximum values will be explained in the dynamic logical partition section.

Each partition whether shared or dedicated has a dedicated quantity of memory assigned to it. Minimum and maximum memory limits are set [IBM2005b]. For POWER5 based systems memory is assigned in blocks of at least 16 MB, although the block size varies, depending on the amount of memory in the system. This is smaller than the minimum 256 MB blocks of POWER4 based systems.

Dynamic Logical Partitioning (DLPAR)

Dynamic logical partitioning (DLPAR) is a term used to describe the addition or removal of resources assigned to a partition without having to shut down or reboot it [IBM2004a]. The topic of dynamic logical partitioning is relevant to the next section on the Partition Load Manager (PLM) as this tool makes extensive use of DLPAR operations. The maximum and minimum values mentioned in the previous section are fixed limits on the quantity of resources which can be assigned to or removed from each partition [IBM2005b]. DLPAR operations are constrained within these limits.

Memory, processors, physical and virtual I/O adapters can be added or removed using DLPAR operations. Currently DLPAR memory operations are not supported by the Linux operating system [BE2004]. A number of additional operations are supported for shared processor partitions [IBM2005b]. These include modifying the number of virtual processors, shared processing entitlement and for uncapped partitions the excess weighting. Partitions can also be switched between capped and uncapped mode.

Partition Load Manager (PLM)

Partition Load Manager makes use of dynamic logical partitioning (DLPAR) operations to manage resource assignment to partitions. It can manage memory and processor resources [IBM2004a]. PLM manages the allocation of entitlements to partitions. It can only be used to manage AIX partitions.

Partitions are organised into one or more groups by PLM [IBM2005b]. For each partition which is to be managed a number of values need to be set for each resource to be managed. These include minimum, maximum and guaranteed resource assignments. Notification thresholds and a weighting are also specified.

There are two types of thresholds, upper and lower. If an upper threshold is reached the partition responsible is marked as a resource requester. Conversely if a lower threshold is reached by a partition it is marked as a resource donor.

For processor resources the CPU utilisation of the partition is used as the measure for the thresholds [IBM2004g]. For memory the utilisation percentage and page replacement rate are used [IBM2004a]. When a partition needs additional resources three sources are checked. First the free pool is inspected to see if there are free resources which can be assigned. If the free pool cannot satisfy this demand, PLM checks if there are any resource donors which can provide the resources. Finally if there are no donors available the shares assigned to the partitions are used to reassign resources. Excess resources are distributed based on the following formula:

Proportion of excess resources to receive = (num shares) / (sum of shares of active partitions in the group) [IBM2004g].

This mechanism is used for managing both CPU memory resources. Where necessary PLM will also change the number of virtual processors [IBM2005b].

PLM will act after a threshold value is reached a number of times [IBM2004g]. The default number is six times, but this value can be set. Samples are taken at ten second intervals. Clearly this results in much higher latencies when managing resources. A delta value setting is used to reassign resources. For memory this is the quantity of memory in megabytes which is to be reassigned. For shared CPU resources this is the percentage by which the resource entitlement should be modified. The default value is ten percent.

PLM can manage both dedicated and shared CPU partitions [IBM2005b], but each group can contain partitions of only one type [IBM2004a].

VMware ESX Server 2.5 (Native-Mode System VMM)

VMware has three server virtualisation systems namely VMware Workstation [SVL2001], VMware GSX Server [VMW2005e] and VMware ESX Server [Wal2002]. ESX Server is the most advanced of the three. This section will focus on ESX server for this reason. Unlike VMware Workstation this system is intended for use in the data centre. As stated previously VMware Workstation is a hybrid virtual machine. It is hosted on a host operating system which provides device driver support which is used for I/O operations. According to [SVL2001] applications in a guest operating system execute directly on hardware until I/O operations need to be performed. The host operating system provides this functionality. In this manner device drivers provided by the host operating system provide compatibility with a wide range of hardware.

ESX Server is a native virtual machine monitor (VMM). As such, it cannot rely on the device driver support of a host operating system. Device drivers are provided by ESX Server. This limits compatibility as fewer devices are supported. The reason for this choice is performance. A Type I VMM does not need to switch between guest operating system instances and a host operating system to provide I/O support. These costly context switches are eliminated. Figure 4.23 depicts guest operating systems hosted on the ESX Server Type I VMM.

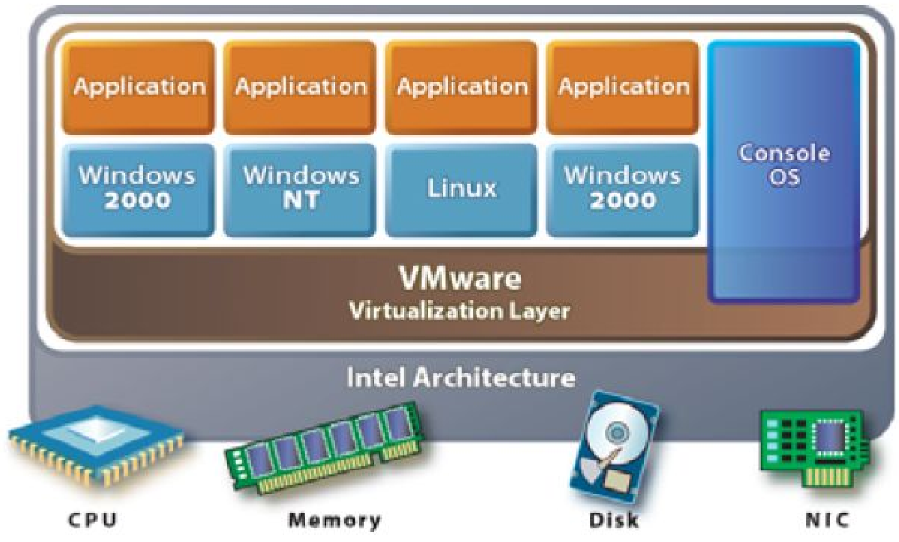


Figure 4.23: The Architecture of VMware ESX Server [VMW2005a]

Implementation Issues and Memory Management

As stated in Chapter 2 the x86 instruction set contains a number of instructions which are not virtualisable. All of the VMware systems use instruction rewriting of these privileged instructions to ensure the integrity of the system. Another challenge faced when hosting multiple guest operating systems is that each is designed to run directly on hardware. Each guest needs to be provided with an address space starting at zero. ESX Server handles this by translating addresses. Memory management in ESX Server is described in greater detail by Waldspurger [Wal2002]. The address space seen by the guest OS is different from the actual hardware address space. Shadow page tables are used to map the one address space onto the other. Another issue is that operating systems typically do not support changes to the quantity of physical memory while running. Each ESX Server virtual machine is provided with a constant physical memory allocation. In practice this quantity can vary using a ballooning mechanism. A balloon module which is installed in each guest operating system instance can be "inflated" to free up memory for other virtual machines. This is depicted in Figure 4.24

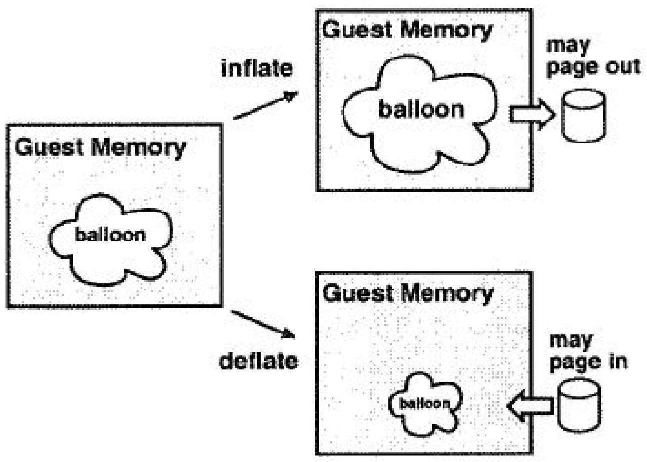


Figure 4.24: Balloon Driver [Wal2002]

The guest OS decides which pages should be swapped out of memory. This is advantageous as the guest can make a more informed choice about which memory should be freed than the VMM. The size of the balloon can be varied to manage the quantity of memory available to the virtual machine. Even though the quantity of memory varies using this mechanism, the guest operating system still believes that it is has the original constant memory allocation.

ESX Server can reduce overall memory usage by sharing pages between partitions. The memory of each VM is scanned searching for pages with identical content. These pages are shared between the guests. This enables the VMM to overcommit memory. The quantity of memory saved depends on the degree of similarity between the applications, data and operating systems of each VM.

Many of the details of resource management by ESX Server are provided in the administration guide for this system [VMW2005a]. Each virtual machine can be assigned minimum and maximum memory allocation values (in megabytes) and a number of memory shares. The ESX Server VMM must ensure that the minimum memory allocation of each virtual machine is always available to it. The maximum value is the quantity of physical memory that the guest operating system believes is present. This value can only be changed when the virtual machine has been shut down.

Shares are used to calculate the quantity of memory to assign to each VM. This actual allocation will also depend on the minimum and maximum limits specified. The actual quantity of memory assigned to each VM is varied using the ballooning technique described previously. Memory which is not actively being used by one VM can be reclaimed for use by others. This is implemented using an idle memory tax [Wal2002]. Virtual machines are charged more shares for memory which they are not actively using. The formula used to calculate the ratio of shares per page is given in Figure 4.25.

$$\rho = \frac{S}{P \cdot (f + k \cdot (1 - f))}$$

Figure 4.25: Memory Tax Formula [Wal2002]

Where S is the number of memory shares held by a virtual machine. P is the number of pages allocated to that VM based on the traditional proportional shares approach, *f* is the fraction of active pages and (1-f) is the fraction of inactive pages. The idle page cost is k, where $k = 1/(1-\tau)$. The tax rate τ is a value such that $0 \leq \tau < 1$. The default value for τ is 0.75. For values approaching 1 all idle memory is reclaimed.

Due to memory overhead not all of the memory in a physical server is available for use by the virtual machines. There are various sources of memory overhead mentioned in [VMW2005a].

- The ESX Server 2.5 virtual machine monitor consumes 24 MB of memory.
- The service console used to manage virtual machines requires 192 MB for eight or fewer virtual machines. This quantity increases as the number of virtual machines grows.
- The VMM ensures that approximately 6% of memory is free at all times to respond to requests for additional memory.
- Data structures and buffers require at least 54 MB per virtual machine. The larger the VM the larger the memory overhead.

This is offset to varying degrees by page sharing between the virtual machines.

CPU Resource Allocation

A virtual machine may have a maximum of two CPUs [VMW2005a]. The Virtual SMP [VMW2004d] add-on to ESX Server is required to enable the use of a second CPU.

Each virtual machine can be assign a minimum and a maximum CPU allocation as well as a number of CPU shares. The minimum and maximum values are specified as percentages. With the Virtual SMP add-on these percentages may be as high as 200% (two CPUs), otherwise the limit is 100% (a single CPU). The minimum value is a guarantee. Even under load the system can provide this processing capacity to the VM. The maximum value is a ceiling on the CPU capacity available to a partition.

As with memory management, shares are used to manage CPU allocations. The CPU resources which can be assigned to a partition are restricted by the minimum and maximum values. Idle processing capacity of one partition can be used by other virtual machines as long as no maximum CPU allocation limits are exceeded.

Network and Disk Bandwidth Allocation

Both network and disk bandwidth allocations can be managed. Network bandwidth is only supported for outbound traffic currently. Network allocation settings include average bandwidth, peak bandwidth and burst size. These can be set for each VM.

Disk bandwidth is managed using shares. Each virtual machine can be assigned a number of disk shares which entitle it to a proportional share of disk bandwidth.

Migrating Virtual Machines

Virtual machines can be migrated to other physical servers. VMware VMotion [VMW2005d] provides this functionality. This migration is useful for responding to variations in service demands [VMW2004c].

Microsoft Virtual Server 2005 (Dual-Mode System VMM)

Virtual Server 2005 is an example of a dual-mode system virtual machine monitor. For more information about this type of virtualisation system refer to Chapter 3. As can be clearly seen in Figure 4.26 Virtual Server 2005 requires a host operating system (Window Server 2003). This is similar to the architecture of VMware Workstation [SVL2001]. The host accommodates multiple guest operating system instances. Virtual Server runs as a service on the host operating system and each virtual machine executes as a thread [MS2004c]. This virtual machine monitor fits between the host operating system and the virtual machines. The host operating system device drivers are used to provide I/O support which occurs in a separate thread for each virtual machine. The lowest block in this diagram indicates that the underlying hardware is x86-based (IA32).

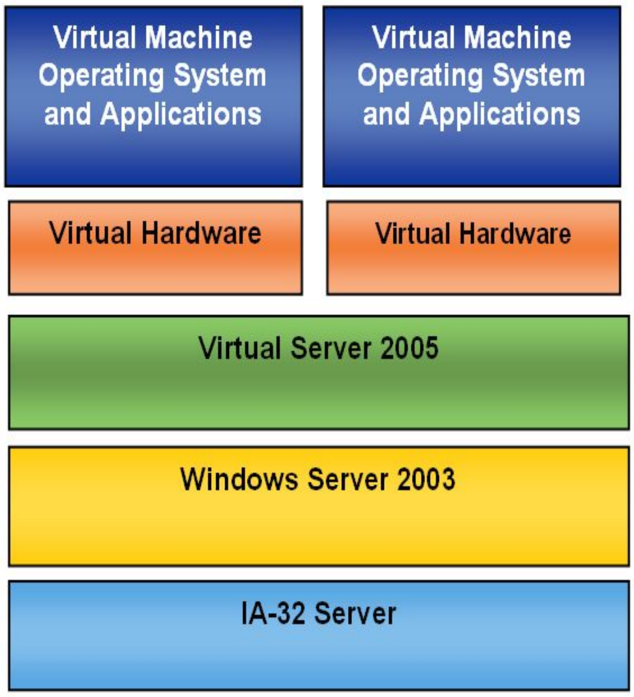

Virtual Machine Operating System and Applications
Virtual Machine Operating System and Applications
Virtual Hardware
Virtual Hardware
Virtual Server 2005
Windows Server 2003
IA-32 Server

Figure 4.26: Architecture of Virtual Server 2005 [MS2004c]

Operating System Support

According to Microsoft [MS2004a] Virtual Server 2005 can host most popular x86 server operating systems. Microsoft only offers support services for Windows Server 2003, Windows Server 2000 and Windows NT 4.0 Server and some variants of these operating systems.

Resource Management

Currently each virtual machine is limited to 3.6 GB of memory [MS2004a] and a maximum of a single physical processor [MS2004c]. This single CPU limit is the same as that of VMware ESX Server without the SMP Server add-on [VMW2004d]. The number of virtual machines which can be hosted on a system depends on the host hardware resources [MS2004c]. To manage CPU resources each virtual machine can be allocated minimum and maximum CPU usage (percentage) limits. Each virtual machine is also assigned a priority between 1 and 10000 which is used as a weighting when handling resource contention between virtual machines. Processor resource allocations can be allocated dynamically. Currently memory resources cannot be allocated dynamically. Each virtual machine can be assigned a subset of the RAM memory of the host system.

Virtual Hard Disks

Virtual machines are assigned storage in the form of virtual hard disks (VHD). A virtual hard disk is stored as a file by the host operating system [MS2004c]. These virtual disks can reside on remote storage systems such as SAN and NAS or local IDE or SCSI disks.

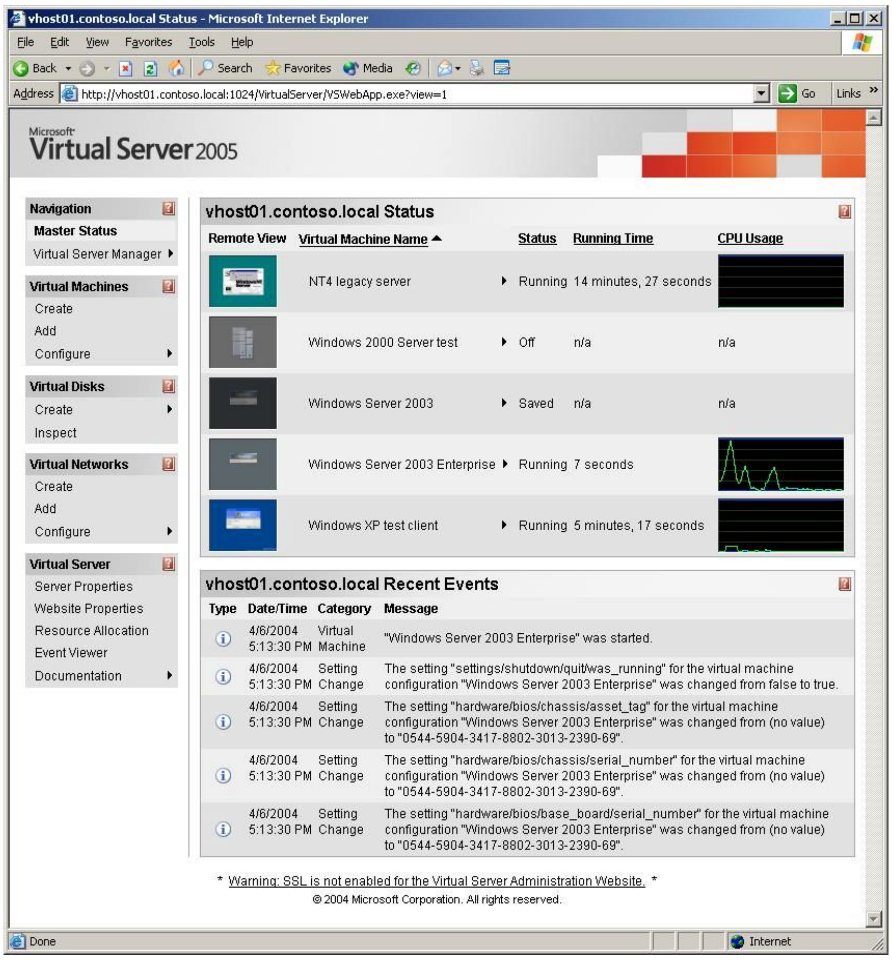
There are various options for creating virtual disks. One option is a differencing disk, which stores changes made relative to a parent disk. Each parent disk can have many children. The parent disk must not be altered, otherwise the contents of any differencing disks relying on it will become invalid. Another option is to create an undo disk which records any changes made to the virtual disks of a virtual machine. An undo disk can be used to restore a virtual disk to a former state. It is also possible for a virtual disk to make use of an entire physical disk. This is referred to as a linked virtual hard disk.

Virtual Networking

Virtual machines can communicate with external networks [MS2004c]. As with all I/O the virtual machines rely on the host to provide this functionality. Virtual networking can be used to connect virtual machines on the same system without relying on physical networking facilities. This network is implemented by copying data in memory rather than transmission of packets. Communication with the host operating system is also possible via a loopback adapter.

Management

Virtual Server exposes a COM interface to enable automated control of the system via external applications. A virtual machine is stored as a virtual hard disk file and an XML configuration file. Virtual machines are managed using a web interface. Figure 4.27 is a screenshot of this interface being used to manage multiple Windows operating system guests.



vhost01.contoso.local Status

Remote View	Virtual Machine Name	Status	Running Time	CPU Usage
	NT4 legacy server	Running	14 minutes, 27 seconds	
	Windows 2000 Server test	Off	n/a	n/a
	Windows Server 2003	Saved	n/a	n/a
	Windows Server 2003 Enterprise	Running	7 seconds	
	Windows XP test client	Running	5 minutes, 17 seconds	

vhost01.contoso.local Recent Events

Type	Date/Time	Category	Message
ⓘ	4/6/2004 5:13:30 PM	Virtual Machine	"Windows Server 2003 Enterprise" was started.
ⓘ	4/6/2004 5:13:30 PM	Setting Change	The setting "settings/shutdown/quit/was_running" for the virtual machine configuration "Windows Server 2003 Enterprise" was changed from false to true.
ⓘ	4/6/2004 5:13:30 PM	Setting Change	The setting "hardware/bios/chassis/asset_tag" for the virtual machine configuration "Windows Server 2003 Enterprise" was changed from (no value) to "0544-5904-3417-8802-3013-2390-69".
ⓘ	4/6/2004 5:13:30 PM	Setting Change	The setting "hardware/bios/chassis/serial_number" for the virtual machine configuration "Windows Server 2003 Enterprise" was changed from (no value) to "0544-5904-3417-8802-3013-2390-69".
ⓘ	4/6/2004 5:13:30 PM	Setting Change	The setting "hardware/bios/base_board/serial_number" for the virtual machine configuration "Windows Server 2003 Enterprise" was changed from (no value) to "0544-5904-3417-8802-3013-2390-69".

* Warning: SSL is not enabled for the Virtual Server Administration Website. *

© 2004 Microsoft Corporation. All rights reserved.

Figure 4.27: Virtual Server Management Interface [MS2004c]

4.3 Conclusion

Understanding virtualisation systems and their capabilities is critical for learning about the current state of the art in server virtualisation. Together with Chapter 3, this literature study provided a clearer picture of the relative strengths and weaknesses of different virtualisation systems and the approaches taken. This information will be used in Chapter 5 to inform the selection of criteria for the framework.

For the literature study presented in this chapter, information had to be gathered from numerous sources. Vendor websites provide an abundance of marketing material. As an objective study this chapter focused on the technical information. Research papers, white papers and lengthy product manuals were examined. Relevant information was often found deep in lengthy technical documents and even in footnotes. The majority of information in this chapter was gathered from system administration guides. These guides proved to be a rich source of information about the features provided by these systems. Unlike virtualisation systems originating in academia, few papers have been published on commercial virtualisation systems. A clear exception is IBM, whose researchers have been publishing papers about virtualisation systems since the 1960s.

Chapter 5: Virtualisation Systems Comparison Framework

5.1 Introduction

The framework which is the subject of this research is presented in this chapter. The design of this framework is based on the literature study conducted in Chapters 2, 3 and 4. To set the scene, some additional background information is discussed first.

There is a wide variety of virtualisation systems. These range from research projects such as UMLinux [HBS2002] and Denali [WSG2002a] to those which are used in commercial environments. The systems covered in Chapter 4 are all examples of the latter.

In order to determine the capabilities of these systems information has to be collected from many sources. It is often necessary to examine research papers, white papers, vendor websites and lengthy product manuals. Other challenges obtaining this type of information include:

- Difficulty in locating some documentation.
- Available information lacks structure
- Relevant information dispersed through numerous sources, requiring synthesis.
- Differences in nomenclature and definitions of concepts across related documents.
- Ambiguous and sometimes conflicting information, requiring critical evaluation.

This makes it difficult for researchers and practitioners to learn about the state of the art in server virtualisation systems.

As far as the author is aware there is currently no independent framework for evaluating server virtualisation systems. Existing comparisons such as those by the Edison Group [EG2004], IBM [IBM2002a] and Sun [SM2004a] available from vendor websites emphasise strong points of vendor systems compared to competing systems.

It would help researchers and practitioners if there was a standard set of criteria for evaluating server virtualisation systems. Information from evaluations using such a set of criteria could be gathered together in a structured form. A framework would satisfy these requirements, because, according to the Federal CIO Council [FCC1999] a framework is defined as *'a logical structure for classifying and organizing complex information'*. The framework proposed in this chapter is used to classify and structure a set of evaluation criteria for server virtualisation systems and the information obtained from evaluations.

The only independent comparison of server virtualisation systems that the author is aware of is a table found in a related work section in a paper by Reumann *et al.* [RMS+2000]. This table compares the resource controls provided by eight *"Resource- and service-oriented server management solutions"*. These range from schedulers and resource management tools to server virtualisation systems. Only three of these systems are widely used. The eleven criteria provided are mostly concerned with resource management. Articles in the IT press such as [IN2004] and [CN2004], although independent, are insufficiently detailed for any meaningful comparison between server virtualisation systems.

This following topics are covered in this chapter:

- Description of intended audience and applications of the framework.
- The structure of the framework and the development of a ranking system
- The framework criteria and the selection process
- Interpreting results

- Defining the concept of a virtualisation requirements filter and discussing related applications

5.2 Intended Audience

Researchers

Researchers implementing server virtualisation systems could use the framework criteria to evaluate their systems. This would enable them to compare their systems to other systems which have been evaluated using the same criteria. The framework could be used to identify areas for improvement. When designing a new virtualisation system, researchers could consult the framework to help them select features to implement and learn about other virtualisation systems.

The potential of the framework is not limited to those researchers who are implementing virtualisation systems. Independent researchers could use the framework to conduct independent evaluations of systems. These evaluations could be submitted for inclusion in the body of knowledge. A researcher who is new to the field could use the framework to learn about the functionality provided by different systems.

Vendors

Vendors could use the framework in a similar manner to researchers. Evaluations of a vendor's virtualisation system(s) conducted by an independent researcher using the framework could be used to identify the strengths and weaknesses of these system(s). This information could be used to identify areas for improvement for future releases. An evaluation of a new system could be conducted internally, prior to release to understand how it compares to competitors' systems.

Data Centre Managers

Decision-makers such as data centre managers could use the body of knowledge structured around the framework to understand the relative strengths and weaknesses of the systems available. This information could be used to identify systems which are best suited to their individual requirements. Decision-makers can define virtualisation requirements filters, consisting of a set of rules based on framework criteria, to specify requirements. These rules can then be used to "filter" a set of available systems to identify suitable those which are the most suitable. Virtualisation requirements filters are discussed later in this chapter.

5.3 Applications

The following applications have been identified based on the preceding discussion:

- Evaluating a new server virtualisation system in order to compare it to existing systems
- Identifying comparative strengths and weaknesses of current systems
- Identifying areas for improvement for an existing server virtualisation system
- Guiding requirements definition for a new server virtualisation system
- Using a virtualisation requirements filter to formally define requirements for a given user's environment
- Using this type of filter to identify suitable virtualisation systems to facilitate decision-making
- Using the framework and filters as the foundation for a decision support tool
- Structuring information about virtualisation systems in a standard format

5.4 Clarifications

Before presenting the framework structure and criteria, two points need to be clarified:

A Note on Terminology

Throughout this chapter the term "partition" will be used as a generic term to refer to any type of virtual server abstraction. This includes virtual machines, operating system containers and physical partitions. These approaches to virtual server abstraction were examined in Chapter 3. The term partition was selected because it is more generic than the term "virtual machine". This is because the term "virtual machine" is not commonly used to refer to container-based systems and physical partitioning systems. This was discussed in Chapter 3.

The use of the term partition in this context is not to be confused with hard disk partitions! In some cases when discussing hard disks the term "virtual server" will be substituted to avoid confusion.

Comparison vs Classification

The purpose of the framework is to evaluate server virtualisation systems in order to compare them using criteria which are relevant to data centre environments. This is not to be confused with classifying virtualisation systems into a taxonomy like the one presented in Chapter 3.

When examining approaches to server virtualisation in Chapter 3 and current virtualisation systems in Chapter 4, the wide variety of implementation approaches and variations thereof became apparent. Although the implementation details of each system will affect its functionality, the focus of the framework is on the features provided by each system, rather than how this functionality was achieved. The incorporation of this type of information would have resulted in an overlap between framework criteria. The inclusion of low-level implementation details would also have resulted in an overly complex framework lacking generality.

The preceding discussion can be summed up with the following statement:

The focus of the framework is on characteristics which influence the practical applicability of virtualisation systems in data centres rather than low-level implementation details.

5.5 Framework Criteria Selection

The framework criteria were selected based on characteristics which are desirable for common applications of virtualisation technology in commercial data centre environments. These applications include virtual web hosting, data centre consolidation and new system testing and transition. These applications were identified in Chapter 3 and correspond to the Internet and organisational data centre types identified in Chapter 2. For additional information about the applications of virtualisation in the data centre refer to Chapter 3.

The framework criteria were selected based on the literature. In Chapters 2, 3 and 4 data centres, virtualisation and current virtualisation systems were examined. This information was used to inform the selection of criteria. The high-level framework criteria were selected through a process of inference. While examining virtualisation literature and technical information about current virtualisation systems for Chapters 3 and 4, seven themes emerged. These themes resulted in the seven high-level criteria or categories which form the foundation of the framework. These high-level criteria were further justified by research papers which identified characteristics of virtualisation systems and their applications. These papers are referenced under the relevant framework categories.

The seven high-level categories are depicted in the upper half of Figure 5.1. These categories and the lower level criteria presented in this chapter are contextualised in terms of data centres. This is because the primary application of virtualisation systems is on server systems which are usually deployed in data centres. All of the high-level categories represent desirable qualities for server virtualisation systems in the context of data centres. For an overview of data centres refer to Chapter 2.

The low-level criteria were selected based on research papers, the framework objectives presented in the previous section and the above-mentioned study of existing virtualisation systems. Relevant literature is referenced throughout this chapter. Only characteristics whose usefulness in a data centre environment could be justified from literature were included.

The absence of certain criteria needs to be justified. The reasons why some criteria were not deemed worthy of inclusion in the framework are discussed immediately after the framework criteria are presented.

The structure and components of this framework are discussed in greater detail later in this chapter.

5.6 Structure of the Framework

The framework can be represented in the form of a tree structure as depicted in Figure 5.1. The framework is divided into seven high-level criteria. These high level criteria are further split into one or more levels of additional sub criteria. The invisible leaves represent the most specific criteria and the ancestors the less specific. The framework criteria are discussed in the next section.

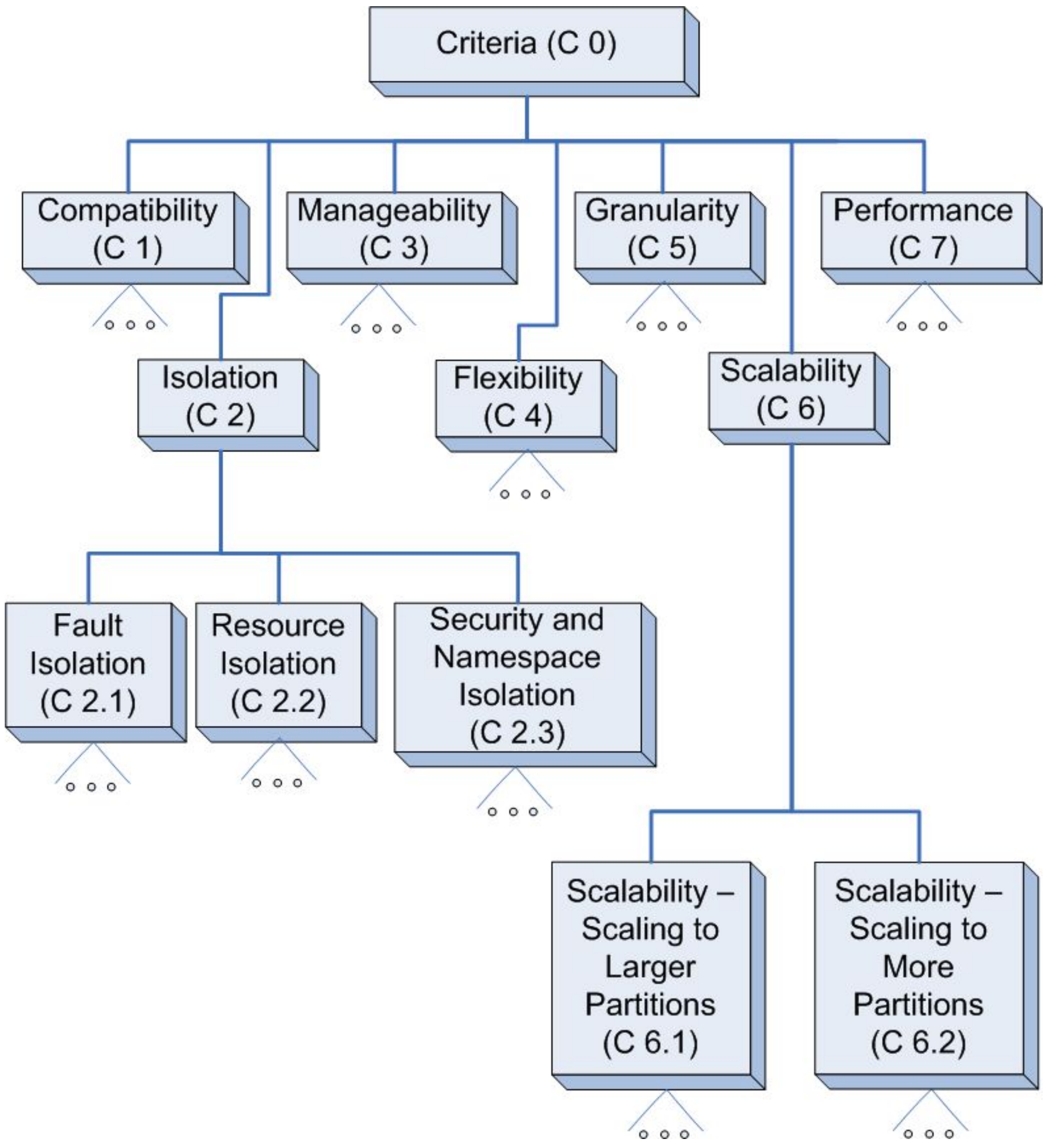

Criteria (C 0)
Compatibility (C 1)
Manageability (C 3)
Granularity (C 5)
Performance (C 7)
Isolation (C 2)
Flexibility (C 4)
Scalability (C 6)
Fault Isolation (C 2.1)
Resource Isolation (C 2.2)
Security and Namespace Isolation (C 2.3)
Scalability – Scaling to Larger Partitions (C 6.1)
Scalability – Scaling to More Partitions (C 6.2)

Figure 5.1: The First Two Levels of the Framework Structure

5.6.1 Structure of Evaluations

The structure of an evaluation using the framework criteria needs to be clearly defined. This is to ensure clarity, and is essential for the virtualisation requirements filter mechanism presented later in this chapter. For each criterion an evaluation consists of two pieces of information, a value and an optional comment.

Value

The value represents the result of evaluating a system using the criterion under consideration. This value needs to conform to the data type specified for that criterion. A data type will be specified for each of the framework criteria presented in the next section. Valid response types are listed in Table 5.1.

Table 5.1: Response Data Types

Data type	*Description*
Boolean	Valid values for this data type include “Yes”, “No” or N/A (not applicable). An example of a criterion with this data type is scripting support. Either a system supports scripting support or it does not. “Yes” and “No” are used instead of true/false to improve readability.
Numerical range	A numerical range consists of an upper and a lower bound e.g. 16-24. A value of N/A (not applicable) is also permissible where appropriate. Values need not be integers. Floating point bounds are also permissible e.g. 0.45-3.25. This data type is especially useful for responses which depend on the configuration of a system. A single number is also a valid value for this data type. In such cases this value is taken as both the upper and lower bound.

Data type	***Description***
List	The list type can be used for criteria which require multiple responses. An example of this is operating systems supported. Responses are provided in a comma-separated list e.g. FreeBSD, Windows, Linux. A list containing a single element "none" can be used to represent an empty list. Duplicate values are not permitted.
Number	The number data type can be used to represent a single numerical value, positive or negative. This value can be an integer or a floating point value.

Additional response types could be added in future if required due to the addition of new criteria.

Optional Comment

A comment can be used to provide additional information. Examples include information about optional settings, system configurations or any other clarifications.

For examples of evaluations conforming to the valid response types refer to Chapter 6.

5.7 Ranking System

Identifying the relative strengths and weaknesses of virtualisation systems is one of the applications of the framework. Initially no attempt was made to rank the systems within each leaf criterion or non-leaf criterion (category). Interpretation was subject to inspection of the evaluation data for each criterion. In order to facilitate comparison a ranking system was adopted for each leaf criterion. This was relatively straightforward once data types were defined for each of the criteria. Systems with more favourable values for a given criterion are ranked above those with less favourable values. In most cases the ordering is obvious. If the approach to ranking systems for a given criterion is not obvious the ranking process will be described for that criterion. A ranking is represented by a number, where smaller numbers indicate a better ranking position. In some cases there will be a tie between two or more systems when ranking. Thus it is possible for more than one system to have the same ranking position for a given criterion.

Ranking the systems using a specific criterion is relatively easy. Constructing a ranking system for a category spanning multiple criteria is a more complex task. Different criteria within a category may have multiple incompatible data types. A mixture of lists, booleans and numerical ranges makes these values difficult to compare directly. Another concern was that representing a category as a single ranking value would be an over-simplification.

Due to these considerations the author was initially reluctant to adopt any kind of per-category ranking system. In order to overcome the challenge of multiple data types a ranking system based on the rankings for each of the criteria within a category was adopted. As rankings are all numerical values this solved the problem of multiple data types. For simplicity each of the criteria within a category are weighted equally. This is a reasonable simplification. The framework criteria within categories do not have any clear order of importance. Criteria correspond to different resource types, types of compatibility or similar concepts where equal priority is a reasonable assumption. Where necessary multiple similar criteria were condensed into a single criterion. This was done to avoid such criteria having a disproportionate influence on the results given the equal weighting system adopted.. This also simplified the framework by reducing the number of criteria.

Developing an equally weighted ranking system for categories proved more challenging than anticipated. Simply taking the average ranking for each system within a category is insufficient. Consider the example of a leaf criterion where there are a large number of systems tied for ranking positions. If there are no ties for another criterion then that criterion will have a larger influence on the average ranking. This is because a large number of ties results in a lower average ranking for a criterion. A solution considered was to weight the criteria based on the number of ties. A weighting multiplier is ineffective however, as ties only dilute the rankings of systems which form part of a tie. As a result a weighting multiplier should only be applied to the rankings of systems which are tied. If there are multiple ties within a criterion, more than one multiplier value will be required. A simpler solution which is much easier to explain was adopted. For each criterion points are awarded based on rankings. If **n** systems have been evaluated using the framework first place is awarded **n** points. Second place is awarded **n**-1 points, third place **n**-2, and so on. In the event of a tie the points which would have been awarded for the positions which the tie spans are averaged. This average value is the number of points which is awarded to each system participating in the tie. This is analogous to the way prize money is awarded as some sports events. If there is a two-way tie for second place the prize money for second and third place is combined and divided evenly between the two players. Using these points an equally weighted ranking can be determined for each category. The average number of points awarded to each system in all of the leaf criteria within a category is used to rank the systems within that category. If a value of N/A (not applicable) is recorded for a system for a criterion, then points average calculation for that system will exclude that criterion. Using a points system also makes the data easier to interpret as higher numbers are intuitively interpreted as better.

This ranking system is used to rank the systems evaluated in Chapter 6. The resultant rankings for each category are presented. The calculations are omitted, but can easily be reproduced using the explanation provided in this section.

Before the ranking system was introduced the relative strengths and weaknesses of these systems were identified by examining the data resulting from the evaluations. The per-category rankings were only calculated later. Interestingly the findings resulting from manually inspecting the evaluation data were consistent with the rankings. The purpose of the ranking system is to provide a quick reference for comparing systems. For a more accurate reflection an analysis of the bare evaluation data is recommended. A section on interpreting results is included later in this chapter.

No overall ranking system has been adopted for rankings spanning all of the categories. Such a category-spanning ranking would be near-meaningless. Not only are the categories not directly comparable, but are unlikely to be of equal importance if they were. The relative importance of these categories would be requirements dependent. If users wish to use the framework to select a virtualisation system meeting their requirements, the virtualisation requirements filter mechanism presented later in this chapter is recommended. This approach is recommended because the most suitable system for a given user, such as a data centre manager, will depend on their specific requirements.

5.8 Framework Criteria

Each of the criteria is numbered. This number appears in brackets after the name of each criterion.

5.8.1 Compatibility (C 1)

Compatibility corresponds to one of the “essential characteristics” of virtual machines identified by Belpaire and Hsu [BH1975], namely that the execution environment of the virtual machine must be logically identical to the real machine, enabling software compatible with one to be compatible with the other. Compatibility was also an important reason why a virtual machine approach was taken for an early virtualisation system, namely VM/370 [Cre1981]. Some of the compatibility criteria selected for the framework presented in this chapter relate to software compatibility which is also a characteristic of virtual machines and virtual machine monitors according to Rosenblum [Ros2004]. The importance of compatibility is discussed further with additional references under each of the compatibility criteria.

While not all virtualisation systems are virtual machines in the strictest sense of the word compatibility is clearly important. Current virtualisation systems such as those examined in Chapter 4 are only compatible with certain hardware and operating system combinations. Compatibility is especially important given the heterogeneous mix of hardware and software in data centres [CS2003a]. The compatibility and management issues associated with heterogeneous environments were listed among common data centre problems in Chapter 2. Another problem was high costs. Compatibility with existing hardware and software reduces the need for disruptive changes when introducing a virtualisation system. Thus virtualisation systems which are compatible with existing hardware are of greatest benefit.

The five dimensions of compatibility are shown in Figure 5.2 and presented in the next section.

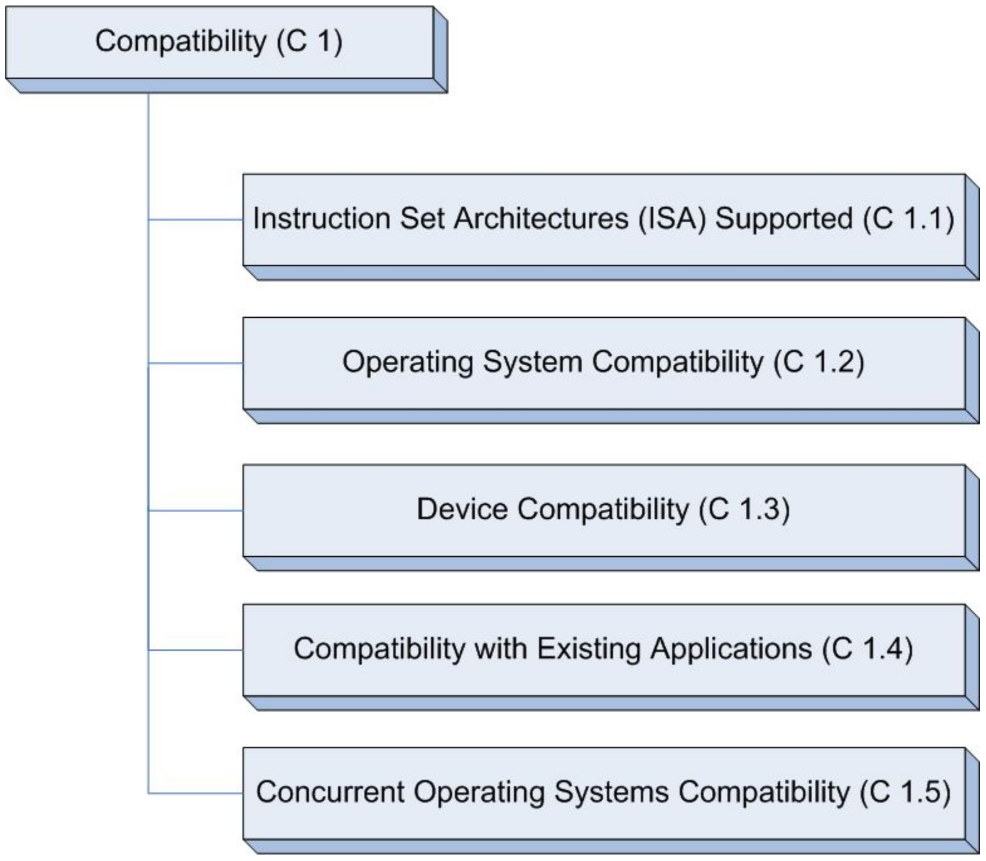

Compatibility (C 1)
Instruction Set Architectures (ISA) Supported (C 1.1)
Operating System Compatibility (C 1.2)
Device Compatibility (C 1.3)
Compatibility with Existing Applications (C 1.4)
Concurrent Operating Systems Compatibility (C 1.5)

Figure 5.2: Compatibility Criteria

Instruction Set Architectures (ISA) Supported (C 1.1)

Heterogeneous data centres contain a wide variety of hardware with different instruction set architectures (ISAs) [CS2003a]. Instruction sets commonly used by servers include x86, x86_64 [AMD2001], POWER [IBM2005d], PA-RISC [HP2005b], SPARC [SI2005] and IA64 (Itanium) among others. Virtualisation systems, such as those covered in Chapter 4, typically support only one or two ISAs.

ISA compatibility is therefore an important requirement if organisations are to use virtualisation with existing hardware. It is also logical that ISA compatibility will influence the selection of new hardware or virtualisation systems.

All ISAs supported by a system should be listed when conducting an evaluation. When ranking systems those which support more ISAs are to be ranked ahead of those which support fewer.

Response Type: List

Operating System Compatibility (C 1.2)

A wide variety of operating systems are used in data centres [CS2003a]. Some virtualisation systems only support a single operating system while others support multiple operating systems. Solaris Containers [TC2004] for example are only supported on Solaris. VMware ESX Server [VMW2005a] is an example of a virtualisation system which supports multiple operating systems. For more details on these systems refer to Chapters 4 and 6.

Some applications are only compatible with a specific operating system. Microsoft SQL Server [MS2005b] for example is only available for Windows. Other applications require a specific operating system version or patch set [HP2004b]. Virtualisation systems compatible with multiple operating systems therefore provide compatibility for more applications. Data centre staff may also have specialised skills in a particular operating system. A virtualisation system which requires switching to another operating system may result in retraining costs.

This criterion is *not* rendered redundant by the subsequent criteria which consider support for multiple operating systems being hosted *concurrently*. To illustrate this point consider the example of Virtuozzo [Vir2005]. This is a container-based system which supports multiple operating systems, but cannot host multiple operating system instances simultaneously unless used in conjunction with another virtualisation system.

Another factor to consider is that some virtualisation systems require custom modifications to be made to existing operating systems. Current examples of systems requiring operating system modifications include the Denali [WSG2002a] and Xen [BDF+2003] paravirtualisation virtual machine monitors. Unless these extensions become standard, vendors are unlikely to support and certify such systems.

All operating systems supported by a system should be listed when conducting an evaluation. For systems which require modified operating system builds, the word "modified" should be included alongside each operating system which requires such a build. An example of this type of response is: Linux (modified), FreeBSD (modified). Information about version numbers can be included in the comment section.

The number of operating systems supported is the measure to use when ranking systems.

Response Type: List

Device Compatibility (C 1.3)

Virtualisation systems such as VMware Workstation [SVL2001] and Virtuozzo [Vir2005] take advantage of the drivers provided by existing operating systems. This provides compatibility with a wide range of devices. For more information on this topic refer to Chapter 3. Other systems such as VMware ESX Server [VMW2005a] provide their own device drivers or modified device drivers. This restricts hardware support to a smaller set of supported devices [VMW2005b]. Compatibility with existing device drivers enables virtualisation systems to be employed on a wide range of systems.

A system which supports at least 90% of the devices supported by an non-virtualised system is considered to satisfy this criterion. Devices not relevant to server systems are not considered. For virtualisation systems designed by vendors to work with their own servers this criterion is satisfied trivially.

Response Type: Boolean

Compatibility with Existing Applications (C 1.4)

Data centres host a wide variety of applications. Examples are listed in Section 2.5. Given the cost and effort of developing such applications it is important that the use of virtualisation does not result in incompatibilities. Most virtualisation systems provide compatibility with existing applications. Some virtualisation systems achieve compatibility with both applications and operating systems by providing a virtual machine interface which appears identical to a real machine. An example of such a system is VMware ESX Server [VMW2005a]. It is also possible to maintain compatibility with existing applications even if there are incompatibilities with existing operating systems [BDF+2003]. Some systems such as Solaris Containers [TC2004] and User Mode Linux [Dik2001b] provide compatibility with the majority of applications, but can be a source of incompatibility in a small number of cases.

Any system which supports the majority (95%+) of existing user applications is considered to satisfy this criterion. Information about exceptions can be included as a comment.

Response Type: Boolean

Concurrent Operating Systems Compatibility (C 1.5)

Most virtualisation systems, with the exception of container-based systems, provide support for concurrent execution of multiple operating system instances. Some of these systems support more than one operating system. Other systems support multiple instances of the same operating system or different versions or patch levels thereof. This is useful because some applications are only compatible with a particular version of an operating system. Running different operating systems or different versions of the same operating system was also listed by Goldberg [Gol1973] as an application of virtual machines. The three types of operating system compatibility will now be discussed. These include support for different operating systems, different operating system versions and different patch levels.

Support for multiple operating systems is particularly important given the heterogeneous mix of hardware and operating systems found in data centres. This was examined in Chapter 2. Examples of operating systems commonly found in data centres include HP-UX [HP2005c], AIX [IBM2005e], Solaris [SM2005b], Linux and Windows [MS2005a] among others. Clearly this needs to be considered when comparing or selecting virtualisation systems. This capability is currently provided by physical partitioning systems such as HP Node Partitions [HP2004i], logical partitioning systems and virtual machine monitor (VMM) based systems such as VMware ESX Server [VMW2005a]. These systems were reviewed in Chapter 4. Virtual machine monitors, logical and physical partitioning were examined in Chapter 3. Container-based systems do not provide this capability, unless used in conjunction with one of the other approaches. For more information about containers refer to Chapter 3.

Some systems do not support more than one operating system, but enable different versions of the same operating system to be hosted simultaneously. Sun Dynamic System Domains is an example of such a system [SM2004f]. This provides compatibility with applications which require a specific operating system version.

Other applications have even stricter requirements and are only certified for a specific patch level. In such cases systems which provide support for different patch levels are more suitable. HP Virtual Partitions [HP2003a] provides support multiple HP-UX instances, each with different patch sets [HP2004k]. Each instance must be the same HP-UX version though [HP2005h].

When evaluating a system using this criterion the levels of operating system compatibility provided by that system are recorded in a list. Valid list elements include “different OS”, “OS version”, “OS patch level” and “none” where OS is an abbreviation for operating system.

If a system supports concurrent execution of different operating systems in different partitions a “different OS” element is added to the list. It should be noted that the term partition here does not refer to a disk partition. Clearly different versions of the same operating system e.g Windows NT and Windows 2003, are not considered different operating systems for this criterion.

An “OS version” element is to be added to the list for a system if it supports concurrent execution of different operating system versions in different partitions. Similarly an “OS patch level” element is to be added to this list if a system must support different patch levels in different partitions simultaneously.

If a system supports none of these features a value of “none” is to be entered in the list.

The following list summarises the relative importance of list elements when ranking systems:

1. Different OS
2. OS version
3. OS patch level
4. None.

Response Type: List

5.8.2 Isolation (C 2)

Isolation is one of the characteristics of virtual machines [Ros2004]. It is not difficult to see that this applies to virtualisation systems in general. The importance of isolation to virtualisation is discussed in greater detail with additional references under each of the isolation sub-criteria. There are three types of isolation considered by the framework. These are fault isolation, resource isolation and lastly security and namespace isolation. Security and namespace isolation is a single category. It could alternatively have been named administrative isolation, but the term security and namespace isolation was selected because it is more descriptive.

Isolation often comes at the expense of granularity [TC2004]. This trade off is a result of the different approaches to implementing virtualisation systems and became apparent when investigating various virtualisation systems in Chapter 4. The different approaches to virtualisation were examined in Chapter 3. Granularity is considered in Section 5.4.5.

The three types of isolation under consideration are depicted in Figure 5.3.

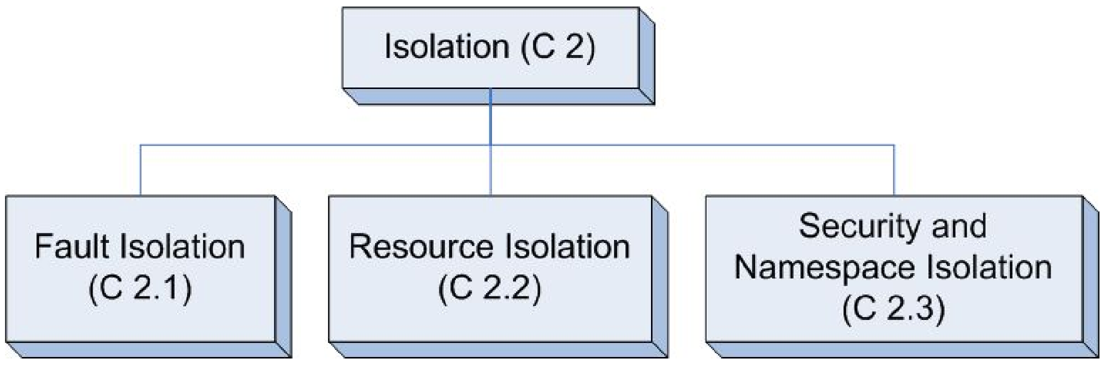

Isolation (C 2)
Fault Isolation (C 2.1)
Resource Isolation (C 2.2)
Security and Namespace Isolation (C 2.3)

Figure 5.3: Types of Isolation

Fault Isolation (C 2.1)

Fault isolation is critically important. This is necessary to ensure high availability of applications. It has long been known that virtual machines can provide software fault isolation [BCG1973]. A contemporary example is that of web applications whose users expect these applications to be available [AID2002]. Availability was identified in Chapter 2 as one of the benefits of data centres [Fir2002]. This is therefore an important requirement for any type of data centre application.

In order to compare the degree of fault isolation of different systems the framework includes criteria focusing on the type of isolation provided by a system. These criteria are presented in the next section and depicted in Figure 5.4.

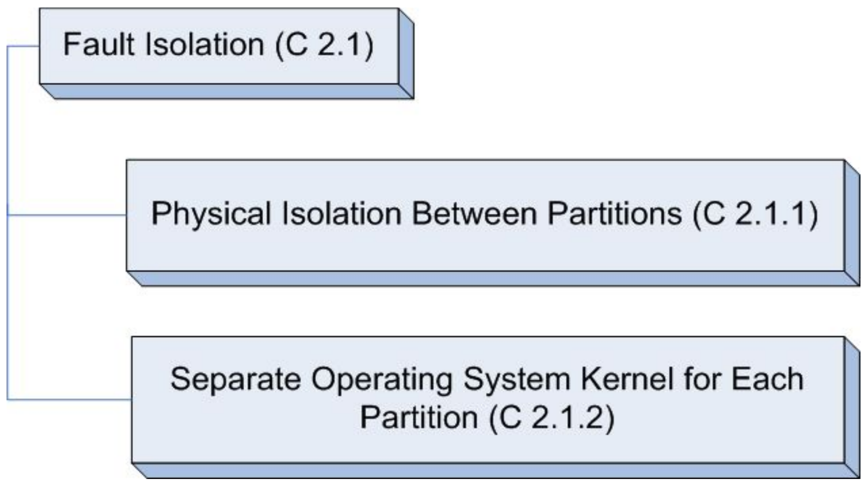

Fault Isolation (C 2.1)
Physical Isolation Between Partitions (C 2.1.1)
Separate Operating System Kernel for Each Partition (C 2.1.2)

Figure 5.4: Fault Isolation Criteria

Physical Isolation Between Partitions (C 2.1.1)

Physical partitioning isolates partitions from hardware failures in neighbouring partitions [HP2004b]. This is a clear benefit over software partitioning. For additional information on physical partitioning refer to Chapter 3. Examples of physical partitioning systems include HP Node Partitions [HP2004i] and Sun Dynamic System Domains [SM2004f]. Cells or system boards are assigned to partitions to ensure that hardware components are isolated.

In order to satisfy this criterion a system must provide each partition with dedicated hardware components providing protection from hardware failures in other partitions.

Response Type: Boolean

Separate Operating System Kernel for Each Partition (C 2.1.2)

Separate operating system instances provide each partition with a separate kernel. Physical partitioning systems such as HP Node Partitions [HP2004i] and Sun Dynamic System Domains [SM2004f] and virtual machine monitor based systems such as Xen [BDF+2003] and VMware [VMW2005a] take this approach. For additional information about virtual machines and physical partitioning refer to Chapter 3.

Software partitioning provides isolation from software faults. It has long been known that virtual machine monitors can isolate systems from software failures, even operating system failures, occurring in neighbouring virtual machines [BCG1973]. This prevents software faults in one operating system instance from propagating to other partitions.

Container-based systems such as BSD jails [KW2000], Linux VServer [LVS2005], Solaris Containers [TC2004, PT2004], Pot Spaces [SHH+2005] and Virtuozzo [Vir2005] share a single operating system kernel and are therefore only isolated from application faults, not operating system faults. Partitions are therefore dependent on the stability of a single operating system instance. The stability of the particular operating system under consideration will have a major influence on the stability of the system as a result. A more detailed discussion of containers is provided in Chapter 3.

The issue of separate kernels is complicated by the fact that many partitioning systems which use separate operating system instances rely on a virtual machine monitor, which is shared between all partitions. As noted by Hewlett Packard [HP2004k] virtual machine monitors (VMMs) and hypervisors are not immune to faults. If the VMM or hypervisor fails, all operating system instances will be affected. More research is needed to determine the frequency of virtual machine monitor faults relative to operating systems faults. Clearly this will depend on the specific operating systems and virtual machine monitors under consideration.

This only requirement for this criterion is that each partition must host a separate operating system instance.

Response Type: Boolean

Resource Isolation (C 2.2)

Another important type of isolation is resource isolation. The resource demands of one partition should not affect the performance of other partitions. This requirement is referred to by Whitaker *et al*. [WSG2002b] and Verghese *et al*. [VGR1998] as performance isolation.

Users have come to expect predictable performance according to Loosley and Douglas [LD1998] and Aron *et al*. [AID2002]. Another expectation of users is that of reasonably small delays according to Shen *et al*. [STY+2002]. If adequate resources are not available server delays can reach unacceptable levels. This is of particular importance to service providers which need to assign resources to different applications based on the relative importance of customers [SS2000]. In these scenarios servers are often shared by multiple customers [RMS+2000]. This corresponds to one of the applications of server virtualisation identified in Chapter 3, namely that of shared hosting by Internet data centres. Improved resource utilisation as a result of sharing was identified in Chapter 3 as one of the advantages of virtualisation. This needs to be balanced against Quality of Service (QoS) requirements [AID2002]. Quality of Service contracts are used to differentiate customer service levels. In order to meet these agreements each partition needs to be isolated from the resource demands of other partitions on the same server. Systems which cannot assign a larger portion of resources to important partitions are less suitable as platforms for service providers as a result.

Isolation of resources is also important for other applications which were identified in Chapter 3 including new system testing and transition and server consolidation. The types of resource controls considered in this category can protect production systems from the resource starvation which could otherwise have occurred as a result of a bug in a new system sharing the same server. For server consolidation projects it is also important that each partition can be provided with a guaranteed quantity of resources. Systems which were once deployed on dedicated servers should not be negatively affected by a move to a virtualised system.

Various approaches have been taken to managing resources. Resources classes are divided by Aron *et al.* [AID2002] into two categories, namely time-scheduled and space-scheduled. Examples of time-scheduled resource classes include network bandwidth, disk bandwidth and CPU time [AID2002]. Examples of space-scheduled resource classes include memory and disk space [AID2002]. Resource scheduling was discussed in Chapter 3 and implemented by the systems reviewed in Chapter 4. One approach proposed by Abdelzaher and Lu [AL2000] is to use a model to manage resource utilisation.

Some systems share resources such as CPUs and network adapters between partitions, while others dedicate resources to partitions with little or no sharing. Some systems provide both options for one or more resources. This can involve trade-offs between isolation and utilisation. Both types will be considered for the framework.

Resource isolation does not only refer to guaranteeing resources for a partition. It can also refer to limiting access to excess resources. At first glance this may appear to run counter to one of the goals of virtualisation systems identified in Chapter 3, namely improved resource utilisation. In some cases, such as managing user expectations for a new system, it is useful to limit resources. Managing user expectations is important because Keil *et al.* [KCL+1998] identified the failure to manage end user expectations as a risk factor for software projects. Although managing performance expectations is only one aspect of managing user expectations, it is relevant. Users of systems expect consistent performance from a system, according to Loosley and Douglas [LD1998]. Response times can become unpredictable if the performance of one partition is dependent on the quantity of resources freed up by other partitions. This variability in response time can therefore lead to negative perceptions of a system. User expectations can be managed by limiting the quantity of resources available to a partition. This application of virtualisation technology is mentioned in virtualisation systems documentation [HP2004b, SM2005d]. Access to excess shared resources can be capped using scheduling techniques. Dedicated resources are implicitly capped. For this reason only capping of shared resources is considered in this section.

The five resource isolation criteria considered for the purposes of the framework are depicted in Figure 5.5. These are discussed further in the next section.

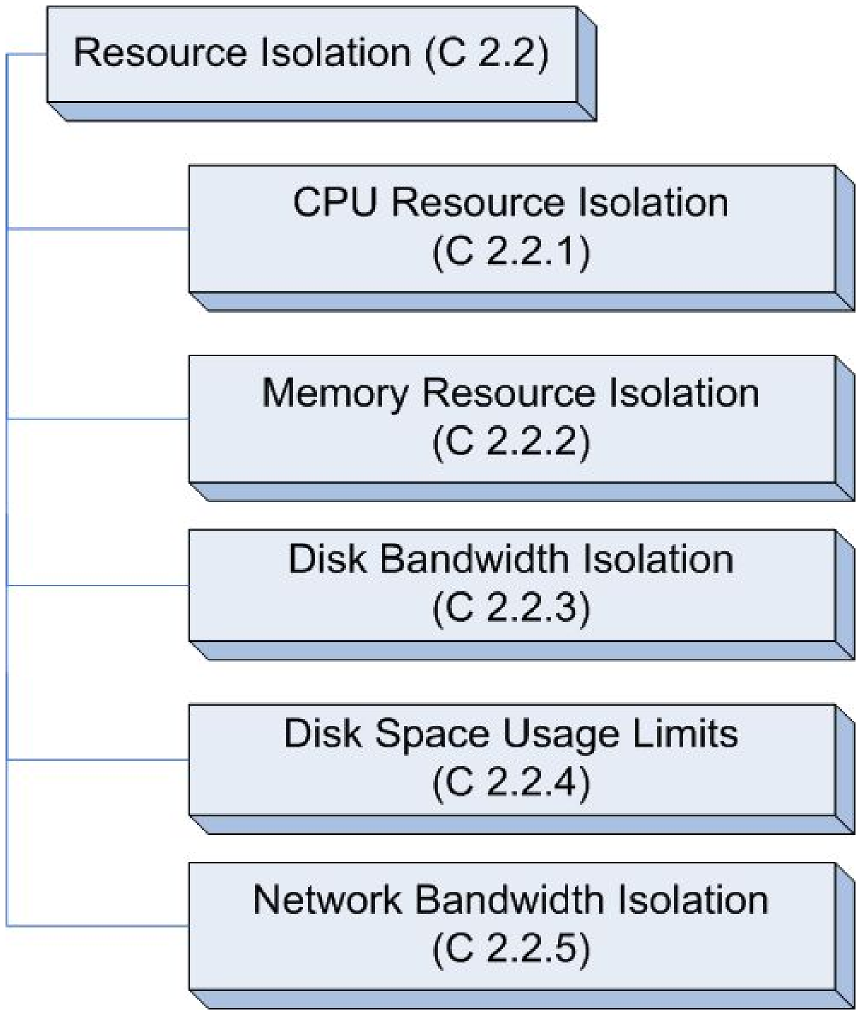

Resource Isolation (C 2.2)
CPU Resource Isolation (C 2.2.1)
Memory Resource Isolation (C 2.2.2)
Disk Bandwidth Isolation (C 2.2.3)
Disk Space Usage Limits (C 2.2.4)
Network Bandwidth Isolation (C 2.2.5)

Figure 5.5: Resource Isolation Criteria

CPU Resource Isolation (C 2.2.1)

The importance of allocating and isolating CPU resources is well known [BDM1999, Wal1995, BGO+1998, AID2002, VGR1998, SS2000]. There are two approaches to providing CPU resource guarantees. These will both be discussed.

The first approach to isolating processor resources is to dedicate one or more CPUs to each partition. This can reduce contention as only processes in that partition are competing for the same CPUs. The alternative approach is to share processors between partitions, but this results in higher cache miss rates [IBM2004f]. Processors dedicated to one partition are therefore better isolated from workload fluctuations in other partitions. These resources are always guaranteed to be available. Systems such as HP Virtual Partitions [HP2003a] and IBM Logical Partitions [IBM2004a] are just two examples of virtualisation systems providing this feature.

The second approach deals with allocation of CPU resources at a sub-CPU level. As discussed in Chapter 2 virtual web hosting, a common application of server virtualisation, involves sharing a single physical server between multiple customers [RMS+2000, WSG2002b]. The number of virtual hosts will usually outnumber the physical CPUs available. Resource isolation is also needed at this level of granularity. The importance of resource isolation was discussed in more detail in the introduction to this section. Processor scheduling can be used to provide resource guarantees for shared CPUs [Wal1995, SS2000, BGO+1998].

When evaluating a system using this criterion the types of CPU resource isolation are to be listed. Valid list elements include “dedicated”, “scheduled” and “unscheduled shared”. A system may support more than one type of isolation.

In order to record a value of “dedicated”, a system must provide support for allocating one or more CPUs to a specific partition. Virtual CPUs are excluded as these may share a physical CPU with other virtual CPUs.

A value of “scheduled” will be recorded if a particular system supports scheduler based proportional allocation of processor resources. The scheduling algorithm used by each system is not considered.

A value of “unscheduled shared” is to be recorded if a system does not provide scheduler-based isolation despite supporting sharing of CPUs between partitions.

If a system supports capping of excess scheduled CPU resources a value of "scheduled capping" is to be recorded. For ranking purposes scheduled capping is considered the least important isolation type. Capping via dedicated CPUs (which is implicit) is already taken into account as dedicated CPUs is the most important type of resource isolation for ranking.

The following list summarises the order of precedence for ranking the various combinations of input. Square brackets indicate optional values. For more information about the ranking system refer to the section on ranking earlier in this chapter.

1. Dedicated [and scheduled] [and scheduled capping]
2. Dedicated and unscheduled shared [and scheduled capping]
3. Scheduled [and scheduled capping]
4. Scheduled capping
5. Unscheduled shared

Response Type: List

Memory Resource Isolation (C 2.2.2)

An important space-scheduled resource is memory [AID2002]. The need for isolating this resource and approaches to achieving this have been explored by a number of sources [SS2000, BGO+1998, Wal2002, PP1973]. This prevents memory leaks in one partition from starving other partitions of memory resources. Memory resource isolation can be achieved either by dedicating a guaranteed quantity of memory to a partition or by enforcing a memory usage cap for each partition. The former approach is more common.

System virtual machines, logical partitioning systems and physical partitioning systems typically allocate a guaranteed quantity of memory to each partition. All of the systems reviewed in Chapter 4 support this feature. The alternative is to enforce a memory usage cap using via a mechanism such as a resource capping daemon. Solaris

Containers [TC2004] (which only provides memory usage capping currently) supports this feature. Due to the asynchronous enforcement of this cap memory consumption can sometimes exceed the cap value.

A system may provide more than one type of memory resource isolation. For this reason the data type for this criterion is a list. Each of the memory isolation approaches supported by a system are to be listed. Valid elements include "guaranteed allocation", "capping" and "none".

If memory can be allocated a partition in such a manner that no other partition is able to access that guaranteed quantity of memory then a value of "guaranteed allocation" is to be added to the list.

A value of "capping" is to be included in the list if a system supports a capping mechanism.

For ranking operations "guaranteed allocation" is considered superior to "capping", which is considered better than "none". This is summarised in the following precedence list (optional values appear between square brackets):

- Guaranteed allocation [and capping]
- Capping
- None.

Response Type: List

Disk Bandwidth Isolation (C 2.2.3)

Disk bandwidth is a time-scheduled resource [AID2002]. Bandwidth guarantees for shared disks have been researched by a number of sources [SS2000, Wal1995, VGR1998, BGO+1998 and AID2002]. This is one of two approaches to providing isolation of disk bandwidth. Both approaches will now be examined. Note that the term "virtual server" is substituted for "partition" in this section to avoid confusing this term with hard disk partitioning.

The first approach, which applies to systems with multiple disks, is to dedicate separate disks to each virtual server. Dedicating a disk to a single virtual server also ensures that the hard disk buffer cache (not to be confused with operating system filesystem cache) is not shared with other virtual servers. By allocating individual disks to virtual servers isolation of disk controllers can also be achieved with some knowledge of which disks belong to which controllers.

Dedicating disks to a virtual server can be achieved on virtually any system with multiple disks, by installing the virtual servers on separate disks. Based on the previous statement one may question why this criterion is included in the framework. The fact that this is available on virtually all systems does not diminish the importance of this feature for isolating disk resources between virtual servers. This criterion provides information about the resource isolation provided by a system. For this reason this criterion is included in the framework.

The second approach is to provide disk bandwidth guarantees for shared disks. This is for cases where disks are shared between multiple virtual servers. In such cases allocating disk bandwidth to virtual servers is critical for ensuring resource isolation, especially for disk constrained applications.

Similarly to the CPU isolation criterion, the response type for this criterion is a list. The types of disk bandwidth isolation provided by each system are to be listed. The isolation types include “dedicated”, “scheduled” and “unscheduled shared”. A system may support more than one type of isolation.

A value of “dedicated” to be included in the list for a system if a disk can be assigned to a virtual server in such a way that no other partition is able to access that disk.

Any system which provides support for guaranteeing disk bandwidth for a shared disk to a virtual server is to have a value of “scheduled” recorded in the list.

If a system supports shared disks but does not provide any scheduling mechanism a value of “unscheduled shared” is to be added to the list.

A value of “scheduled capping” is to be recorded if capping of excess disk bandwidth is supported. Scheduled capping is the least important type of isolation when ranking systems. A separate value for capping via dedicated disks is not included as this is implicit.

Clearly the list for a given system can contain multiple elements.

For ranking purposes the following list summarises the order of precedence for ranking the various input combinations. Square brackets indicate optional values.

1. Dedicated [and scheduled] [and scheduled capping]
2. Dedicated and unscheduled shared [and scheduled capping]
3. Scheduled [and scheduled capping]
4. Scheduled capping
5. Unscheduled shared

This list is the same as for CPU resource isolation.

Response Type: List

Disk Space Usage Limits (C 2.2.4)

Disk space is a space-scheduled resource [AID2002]. Limiting disk space usage can prevent one virtual server from consuming all available disk space and causing other virtual servers to fail due to a lack of disk space. As with dedicating disks to a partition this is possible on virtually any system. A virtual server can be installed within a disk partition or on a dedicated disk. Enforcing disk space limits is important because according to Aron *et al*. [AID2002] service providers often specify disk space quotas in service contracts.

In order to satisfy this criterion a system must provide a means of enforcing disk usage quotas. Systems which only support dedicating disks to virtual servers are considered to automatically satisfy this criterion. For systems which enable disks to be shared between partitions, another mechanism of enforcing quotas must be available to satisfy this criterion.

Response Type: Boolean

Network Bandwidth Isolation (C 2.2.5)

Network bandwidth is a time-scheduled resource [AID2002]. Isolation of this resource is useful for ensuring that Quality of Service (QoS) requirements are met [AID2002, SS2000 and RMS+2000]. Network guarantees can be enforced by dedicating one or more network adapters to a partition or via scheduling for a shared adapter. Both of these mechanisms will now be considered.

Similar to the way dedicated disks provide disk bandwidth isolation dedicating network adapters to a partition results in network bandwidth isolation. This is because partitions with dedicated physical network interfaces do not have to contend for the resources of a shared adapter. HP Virtual Partitions [HP2003a], IBM Logical Partitions [IBM2005b] and Sun Dynamic System Domains [SM2004f] are just a few examples of systems which support this.

Network bandwidth guarantees are necessary to achieve isolation of network bandwidth for partitions which share a physical network interface. Examples of systems supporting network bandwidth guarantees for shared adapters include SODA [JXE2004] and VMware ESX Server [VMW2005a].

When evaluating a system using this criterion, the types of network bandwidth isolation provided need to be listed. If a network adapter can be assigned to a partition in such a manner that no other partition is able to access it, a value of “dedicated” is to be included in this list. If a system provides support for allocating a guaranteed quantity of network bandwidth to a partition when a network adapter is shared between partitions, a value of “scheduled” is to be added to the list. If a system does not provide support for scheduling despite providing support for sharing a network adapter between partitions, a value of "unscheduled shared" is included in the list. A value of "scheduled capping" is to be recorded if a system supports capping of excess shared network bandwidth. As with previous criteria, scheduled capping is the least important type of isolation when ranking systems. Capping provided by dedicated network adapters is not considered separately as this is implicit.

Each system may have multiple values listed. As with CPU resource isolation and disk bandwidth isolation, the following list summarises the order of precedence for the various input combinations for ranking purposes :

1. Dedicated [and scheduled] [and scheduled capping]
2. Dedicated and unscheduled shared [and scheduled capping]
3. Scheduled [and scheduled capping]
4. Scheduled capping
5. Unscheduled shared.

Response Type: List

Security and Namespace Isolation (C 2.3)

Applications of virtualisation identified in Chapter 3 include server consolidation, new system testing and transition and virtual web hosting. Server consolidation and shared hosting are commonly mentioned in virtualisation papers such as [Wal2002, TC2004, RMS+2000, UVS+2004, SHH+2005 and JXE2004]. What these applications have in common is that each partition needs to be isolated from the others, both in terms of security and administration. These two categories of isolation were combined into one category named "security and namespace isolation", as there is an overlap between the criteria used to evaluate them. A number of the security and namespace isolation criteria identified in this section correspond to some of the requirements for application service hosting platforms (ASHP) identified by Jiang *et al*. [JXE2004]. This is discussed further under the relevant criteria.

The importance of security isolation has been known for many years. One of the applications of virtual machines listed by Goldberg [Gol1973] was for securing critical applications. Security isolation also corresponds to one of the essential characteristics of virtual machines identified by Belpaire and Hsu [BH1975], namely that of “impassable walls” between virtual machines. This prevents each system from interfering with the others. More recently security isolation was also listed by Whitaker *et al*. [WSG2002b] as an important requirement. This further highlights the importance of security isolation.

The six dimensions of security and namespace isolation are listed in Figure 5.6 and explained in more detail in the next section.

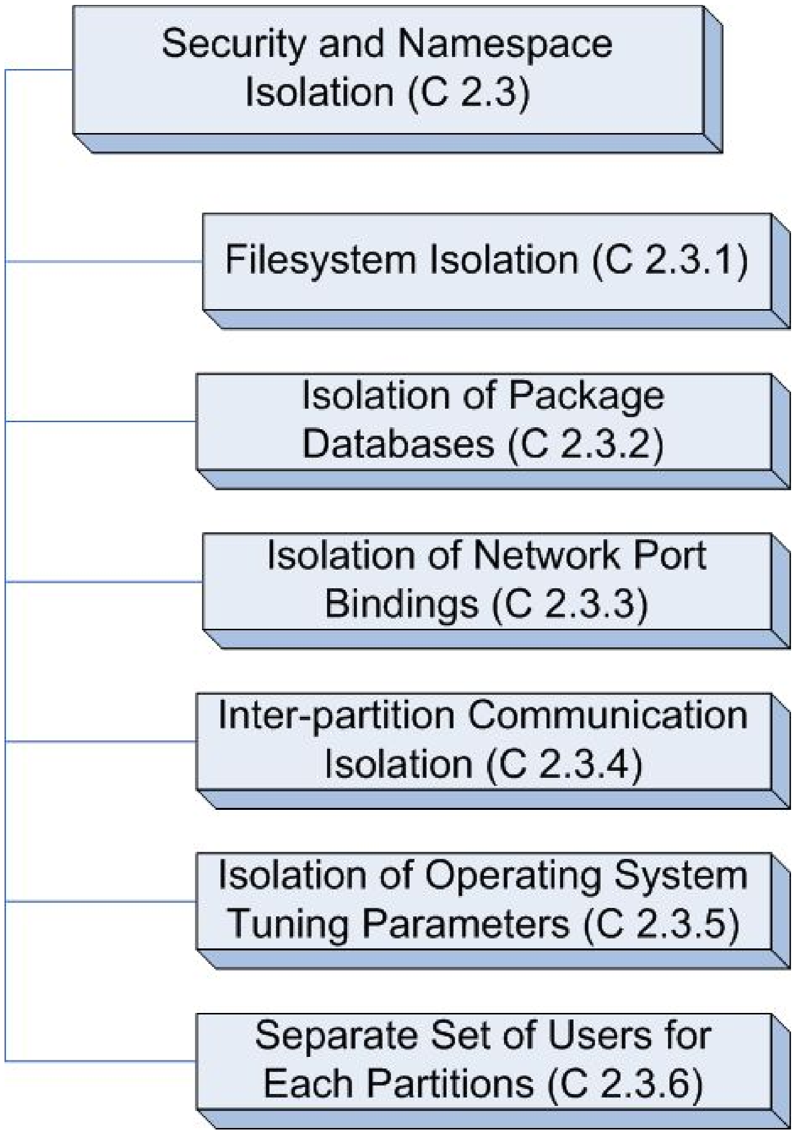

Security and Namespace Isolation (C 2.3)
Filesystem Isolation (C 2.3.1)
Isolation of Package Databases (C 2.3.2)
Isolation of Network Port Bindings (C 2.3.3)
Inter-partition Communication Isolation (C 2.3.4)
Isolation of Operating System Tuning Parameters (C 2.3.5)
Separate Set of Users for Each Partitions (C 2.3.6)

Figure 5.6: Security and Namespace Isolation Criteria

Filesystem Isolation (C 2.3.1)

Server consolidation, shared hosting and new system testing and transition were identified as applications of virtualisation in Chapter 3. For applications such as these each virtual server needs to have its files isolated from others to ensure integrity and confidentiality [KW2000]. This criterion is similar to what Surányi *et al.* [SHH+2005] describes as storage isolation.

Filesystem isolation is also related to the installation isolation criterion mentioned by Jiang *et al.* [JXE2004]. This is because each customer may require different versions of the same file stored at the same location. Thus this criterion is important for both security and namespace isolation.

In order to satisfy the filesystem isolation criterion, a virtualisation system needs to meet two requirements:

- Users of one virtual server should not be able to access the files of another virtual server unless permission to do so has been explicitly granted.
- There must be no directories shared between virtual servers as read only unless done so to ensure the integrity of system settings.

This second requirement is to provide namespace isolation by enabling each virtual server to store a different version of the same file at the same location within each of their respective filesystems. The storage isolation approach of Surányi *et al.* [SHH+2005] does not meet this second requirement.

Copy on write systems are considered to satisfy the second criterion. This approach is used by Microsoft Virtual Server, with a differencing disk [MS2004c]. This also appears to be the approach taken by a project currently in progress, known as the Xen Filesystem or XenFS [Wil2005a]. Read-only loopback mounts are optional for Solaris Containers [SM2005d] which were examined in Chapter 4.

Response Type: Boolean

Isolation of Package Databases (C 2.3.2)

Bearing in mind the applications of virtualisation identified previously, it is clear that different partitions will often need to have different sets of applications installed. Each partition may require a different version of the same library to be installed [JXE2004], which can result in conflicts. Virtualisation can be used to install different versions of the same application on a server without conflicts according to Price and Tucker [PT2004]. This criterion relates to the installation isolation criterion listed by Jiang *et al*. [JXE2004]. In order to satisfy the isolation of package databases criterion it must be possible for each virtual server to have its own unique set of applications installed.

Response Type: Boolean

Isolation of Network Port Bindings (C 2.3.3)

Applications such as web servers need to bind to specific ports. If one application has already bound to a specific port other applications will be unable to bind to it. As discussed in Chapter 2, these types of applications are commonly used by service providers managing Internet data centres and organisational intranet data centres. The need for isolation of network port bindings corresponds to the description of network isolation provided by [SHH+2005] and the installation isolation requirement of [JXE2004]. If no restrictions are in place, an application is free to bind to any port on all available IP addresses. Consider the example of a web server binding to port 80. This would prevent any other applications from using this port. In one example described by Price and Tucker [PT2004], a customer decided to purchase an additional server just to resolve a conflict between two applications which were trying to bind to the same port.

In order to isolate network port bindings each partition needs to be prevented from binding to IP addresses assigned to other partitions. Some partitioning systems isolate port bindings by hosting multiple operating system instances [BDF+2003, VMW2005a and HP2003a]. Other systems such as FreeBSD jails [KW2000], Solaris Containers [TC2004] and Linux VServer [BL2005] use operating system facilities to ensure that partitions can only bind to a single IP address.

Response Type: Boolean

Inter-partition Communication Isolation (C 2.3.4)

The importance of security and the principle of *"impassable walls"* [BH1975] between partitions have already been discussed. Another important security requirement emphasised by Tucker and Comay [TC2004] and Surányi *et al.*

[SHH+2005] is that processes in different partitions should not be able to observe or communicate with each other. This is necessary to ensure that the activities of isolated applications are not visible to each other. Inter-partition communication should be limited to using standard network interfaces. This corresponds to the controlled communication requirement outlined in [JXE2004] and process space isolation described in [SHH+2005].

Any system which prevents processes in one partition from communicating with or monitoring processes in other partitions meets this criterion.

Response Type: Boolean

Isolation of Operating System Tuning Parameters (C 2.3.5)

Modern operating systems provide a number of settings such as kernel tuning parameters [HP2005a]. Different tuning options work better for some applications, and therefore partitions, than others. By isolating tuning parameters, each partition can set parameters to best suit the types of applications which are being run. Systems which host multiple operating systems instances such as Xen [BDF+2003], VMware [VMW2005a] and vPars [HP2003a] provide this functionality without any additional work.

This criterion is met if the system supports isolation of all operating system tuning parameters. If a system only supports isolation of some parameters this criterion is not met. In these cases information about parameters which are isolated can be provided as a comment.

Response Type: Boolean

Separate Set of Users for Each Partition (C 2.3.6)

For applications such as web hosting, service providers use virtualisation systems to provide each partition with a separate set of users. This enables them to provide customers with greater autonomy [KW2000]. Each partition can have its own root user which has administrative authority within that partition.

Partitions are sometimes administered by different departments within an organisation [PT2004]. It is therefore important to have a separate set of users for each partition, providing administrative autonomy to each department. This is especially important for server consolidation where servers with different administrators are consolidated onto a single server.

The need for separate sets of users relates to the administrative isolation requirement listed by Jiang *et al.* [JXE2004].

In order to satisfy this criterion, a virtualisation system needs to provide support for a completely separate set of users for each partition. It must be possible to configure the system in such a manner that users in one partition cannot log into other partitions using the same credentials.

Response Type: Boolean

5.8.3 Manageability (C 3)

One of the common data centre problems identified in Chapter 2 was the high cost of managing systems. Virtualisation systems are intended to improve manageability by replacing physical servers with virtual servers [HP2002]. The ease with which these partitions can be managed by an administrator is therefore an important criterion to consider when evaluating virtualisation systems. Although manageability may influence the total cost of ownership of a system, cost is not considered as one of the framework criteria. The reasons for this exclusion from the framework are explained later in this chapter.

The manageability criteria were selected based on the manageability features and tools available for the virtualisation systems reviewed in Chapter 4. Features offered by other systems not reviewed in Chapter 4 such as Xen [BDF+2003], BSD jails [KW2000] and Linux VServer [LVS2005, BL2005], were also considered, but these systems did not offer any additional manageability features not covered by the other systems. A relatively small set of manageability criteria were selected to keep the framework reasonably concise and avoid giving the framework too much of a commercially oriented flavour. The criteria selected are shown in Figure 5.7 and described in the next section.

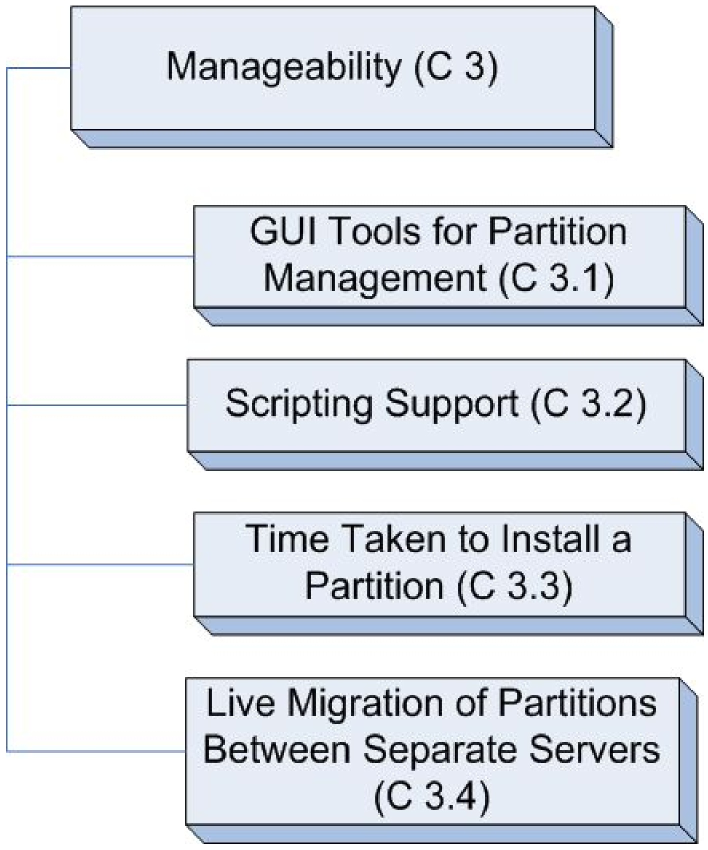

Manageability (C 3)
GUI Tools for Partition Management (C 3.1)
Scripting Support (C 3.2)
Time Taken to Install a Partition (C 3.3)
Live Migration of Partitions Between Separate Servers (C 3.4)

Figure 5.7: Manageability Criteria

GUI Tools for Partition Management (C 3.1)

Virtualisation systems such as Microsoft Virtual Server [MS2004c], VMware ESX Server [VMW2005a] and IBM Logical Partitions [IBM2004a] provide graphical tools for managing partitions. Another system, Solaris Containers [TC2004], originally provided only a command line interface for managing containers. A web interface was later released as a separate product [SM2005a]. Virtualisation systems which are being developed as research projects are unlikely to provide graphical management tools.

Graphical interfaces such as web-based consoles are particularly attractive for inexperienced administrators. More experienced users may not regard this as important. In addition to catering for inexperienced users graphical tools can provide visual feedback about the status of partitions.

Any system which provides graphical tools for configuring partitions (virtual servers) meets this requirement.

Response Type: Boolean

Scripting Support (C 3.2)

Scripting support enables partition management tasks to be automated. VMware ESX Server [VMW2005a] and Microsoft Virtual Server [MS2004c] for example provide scripting APIs. Other systems such as Solaris Containers [TC2004] and HP Virtual Partitions [HP2003a] can be scripted by using Unix shell scripts. Scripting enables a complex series of commands can be saved as a script for future use, thereby providing improved manageability. Administrators can also configure other tools to call these scripts.

This criterion requires a virtualisation system to support either a specially exposed scripting interface or management via standard shell scripts.

Response Type: Boolean

Time Taken to Install a Partition (C 3.3)

Installing software on a new system can be a time-consuming process. If installing a new partition on a system is time-consuming for administrators this will be a manageability burden. This time can become significant if a large number of partitions need to be installed on a system. This is particularly relevant for applications such as virtual web hosting [SHH+2005], which are likely to require a large number of partitions. The time taken to install a partition using a given virtualisation system will be dependent on the specific hardware and operating system used.

Five partitions are to be created, with the time taken to install each timed individually. There should be no delay period between the creation of each partition. The system should be reset before the test commences. The mean of the install times is to be recorded, along with relevant system configuration details. An example of such an evaluation is contained in Chapter 7.

The response type for this criterion is specified as a numerical range, because the time taken to install a partition may depend on factors such as the software or hardware configuration. The upper and lower bounds of this range are used to record the longest and shortest times taken to prepare a partition respectively. The times are to be measured in seconds. Any relevant information about the configurations used to record these times needs to be included in the comments section.

Response Type: Numerical range

Live Migration of Partitions Between Separate Servers (C 3.4)

Live migration entails migrating a virtual machine (or other partitioning abstraction) from one physical server to another without terminating the execution of applications within that partition. According to Clark *et al*. [CFH+2005] live migration is a useful tool. This source describes a number of applications. If one server needs to be serviced, the partition can be moved to another machine. Another option is to migrate a partition to a larger server if demand for resources exceeds the capacity of the current server. According to this source [CFH+2005] these features improve manageability significantly. Another example of a live partition migration tool is VMware VMotion [VMW2005d]. Currently most virtualisation systems do not support this feature.

Response Type: Boolean

5.8.4 Flexibility (C 4)

For the purposes of the framework, flexibility may be defined as *the ease with which a system can be adapted to meet changing resource requirements.*

It is well known that service demands on web applications can vary unpredictably [CGS2003, WSG2002b, STY+2002]. Flash crowds can result in sudden spikes in traffic to a website. Fluctuations also occur in demand for other application services such as those hosted by organisational data centres. Traffic patterns of organisational applications often depend on business hours. Organisational data centres were discussed in Chapter 2. According to [SS2000] even if the resource requirements of an existing application are already known, these will often change. This is partially related to the problem of adapting to changing requirements identified in Chapter 2.

Given the fact that users have come to expect predictable response times [AID2002] fluctuations in demand for resources need to be met gracefully.

The three flexibility criteria are depicted in Figure 5.8 and are discussed in the next section.

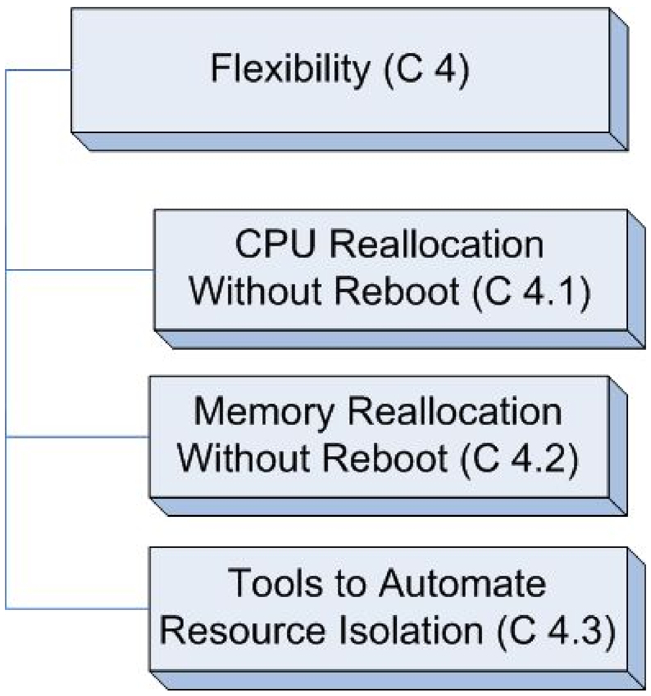

Flexibility (C 4)
CPU Reallocation Without Reboot (C 4.1)
Memory Reallocation Without Reboot (C 4.2)
Tools to Automate Resource Isolation (C 4.3)

Figure 5.8: Flexibility Criteria

CPU Reallocation Without Reboot (C 4.1)

Management of CPU resources is critical for many server applications. Migration of CPUs between partitions without rebooting in response to demand or policy changes results in greater flexibility. Virtualisation systems such as IBM Logical Partitions [IBM2004a], Solaris Containers [SM2005d] and Sun Dynamic System Domains [SM2004f] support reallocation of CPUs without rebooting. This is also known as dynamic CPU migration [HP2004b]. For more information on these systems refer to Chapter 4. If a partition (or the entire server) has to be rebooted in order to migrate one or more CPUs between partitions this results in downtime and therefore limits the flexibility of the system. This criterion considers reallocation of either physical or virtual CPUs.

Response Type: Boolean

Memory Reallocation Without Reboot (C 4.2)

This criterion is similar to the previous one. The fact that resource demand of applications change over time, often unpredictably, was noted in the introduction to this category. Reallocation of memory is useful for dealing with changes in demand for memory by partitions. Memory needs to be reallocated in sufficient quantities to where it is needed.

One of the challenges facing those implementing virtualisation systems is that some operating systems do not support dynamic memory reallocation. An example cited by [BE2004] is Linux. Consequently some systems such as VMware ESX Server [VMW2005a] and Microsoft Virtual Server [MS2004c] do not support this feature.

In order to satisfy this criterion the quantity of memory as observed by the software in the affected partitions needs to change. Changing the quantity of physical memory available to partitions using ballooning mechanisms is excluded as this is only a partial solution. The ballooning mechanism was discussed in Chapter 4 in the section on VMware ESX Server. Rebooting of partitions should not be required to perform a reallocation.

Response Type: Boolean

Tools to Automate Resource Allocation (C 4.3)

The unpredictable nature of server loads make automating resource allocation policies appealing. Some virtualisation systems provide automated resource reallocation functionality to perform this task. There are two approaches to automated resource management tools. Each of these approaches will now be discussed.

The first approach involves the use of system goals. Examples of system level goals include utilisation, locality or simply matching resource supply to demand. Solaris Resource Manager which is a component of Solaris Containers [SM2005d] is an example of a system providing this type of functionality. Partition Load Manager (PLM) for IBM LPARs [IBM2004a] is another example. Refer to Chapter 4 for more information about these systems.

Another approach to managing resource allocation is to use application level metrics. The server applications themselves provide feedback about performance. This approach was used by Aron *et al.* [AID2002] for managing quality of service (QoS) levels for web applications sharing the same server. An example of another such system is HP Workload Manager [HP2004d]. For more information on this system refer to Chapter 4. Examples of data centre goal metrics include transactions per second throughput for a database or web page views per second for a web application. Additional support for specific applications is required to gather metrics [HP2004g].

A list of all automated resources management approaches provided for a given system should be listed. Valid list elements include “system goal”, “application goal” or “none”. A system goal element is added to the list, when evaluating a system, if that system provides a mechanism to automate reallocation of resources from one partition to another in response to system goals. An application goal value is to be included in the list if the system provides a mechanism to automate reallocation of resources from one partition to another in response to feedback from instrumented applications meets this criterion. A response of “none” is entered if a system does not provide either of these mechanisms.

A system may support both application and system goal based approaches. For ranking purposes systems which support more approaches are to be ranked higher.

Response Type: List

5.8.5 Granularity (C 5)

Granularity in this context refers to how finely resource allocations can be managed between partitions. For this framework category the granularity classification use by Chandra *et al*. [CGS2003] was adopted. According to this source there are two types of granularity, namely spatial granularity and time granularity. Time granularity refers to the time taken to reassign resources. Spatial granularity refers to the quantity of resources such as memory and CPUs that can be reassigned from one service to another. Figure 5.9 clearly illustrates the relationship between granularity and how finely resources can be matched to application demand.

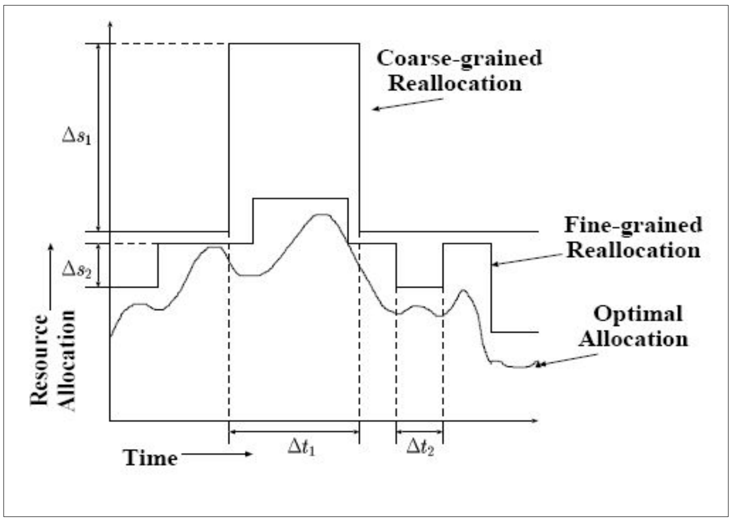


Figure 5.9 : Resource Allocation Granularity [CGS2003]

During the examination of current virtualisation systems presented in Chapter 4 it became clear that granularity and isolation are often conflicting goals. Physical partitioning systems such as [HP2004i] and [SM2004f] typically offer isolation at the cost of granularity. Conversely, systems such as Virtuozzo [Vir2005] and Solaris 10 Containers [TC2004] can partition even small servers into many partitions, but provide lower levels of isolation.

In the granularity category of the framework there are four criteria – two time granularity criteria and two spatial granularity criteria. To keep the framework concise, only CPU and memory resources are considered. The four granularity criteria are shown in Figure 5.10 and explained in the section which follows.

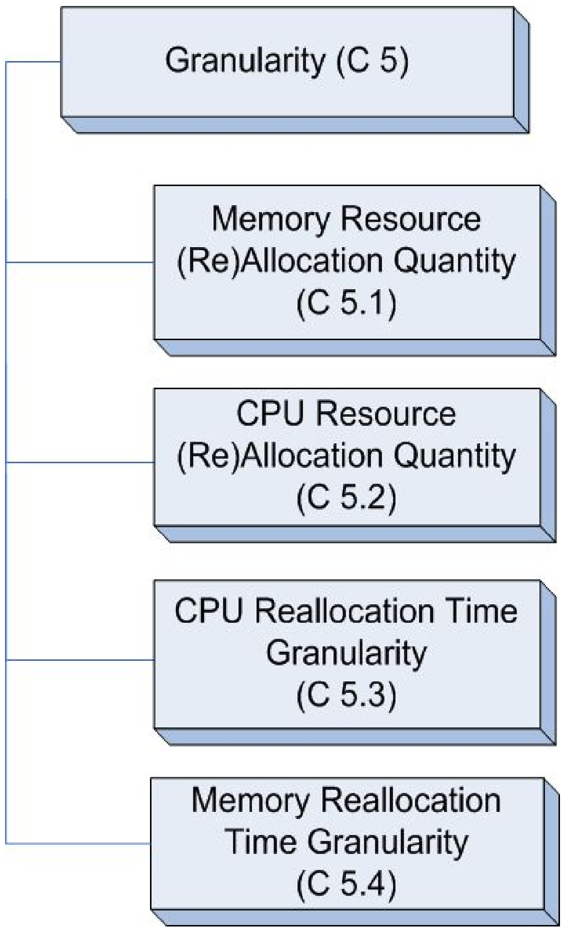

Granularity (C 5)
Memory Resource (Re)Allocation Quantity (C 5.1)
CPU Resource (Re)Allocation Quantity (C 5.2)
CPU Reallocation Time Granularity (C 5.3)
Memory Reallocation Time Granularity (C 5.4)

Figure 5.10: Granularity Criteria

Memory Resource (Re)Allocation Quantity (C 5.1)

It should be clear from Chapter 4 that most virtualisation systems provide support for dedicating quantities of memory to specific partitions. This was one of the resource isolation categories, namely memory isolation. The spatial granularity of memory allocations is now considered. The reason for the title of this criterion will now be explained. For systems which support dynamic memory reallocation, the granularity of memory *re*allocations is considered. For systems which do *not* support dynamic memory reallocation, the granularity of memory allocations is considered. Thus when evaluating memory granularity it does not matter whether changing the memory allocations is dynamic or not, as this was already considered in the flexibility category.

When memory is allocated to a partition or reallocated between partitions, there are sometimes limits on the granularity of these allocations. The quantity of memory which may be (re)allocated may have to be a multiple of a specific value, or can be dependent on the physical server hardware. In the case of physical partitioning systems such as HP Node Partitions [HP2004i] and Sun Dynamic System Domains [SM2004f] the finest level of granularity is to reallocate all of the memory on a system board from one partition to another. For other systems such as virtual machine monitor based systems, the memory granularity may depend on the amount of memory in the system. Therefore this value may not be constant for a given virtualisation system. An example of a system where memory reallocation spatial granularity depends on the amount of memory in the system is IBM Logical Partitions [IBM2005b].

The response type for this criterion is specified as a numerical range. This is because the memory granularity may depend on the system configuration as discussed in the previous paragraph. The lower bound of this range represents the finest memory granularity available in any configuration. If the finest granularity in another configuration is coarser, this should be recorded as the upper bound. Relevant information about the configurations which provide these levels of granularity is to be recorded in the comment section. The only variations in configuration to be considered are for physical partitioning systems, where different servers may have different per-board memory limits. Memory granularity for physical partitioning systems is to be measured using fully configured system boards using the maximum DIMM module size.

When ranking systems the lower bound is to be used.

Response Type: Numerical range

CPU Resource (Re)Allocation Quantity (C 5.2)
Similar to the previous criterion, the spatial granularity of CPU resource allocations is now considered. The explanation of this criterion is similar to that of the previous criterion. The granularity of *re*allocations is considered for systems which support dynamic CPU reallocation. If this feature is *not* supported, the granularity of allocations is to be considered. This ensures generality, as systems which do not support dynamic CPU reallocation can also be evaluated.

The quantity of CPU resources which can be reallocated from one partition to another depends on the virtualisation system being used, its configuration and possible hardware factors. VMware ESX Server [VMW2005a] and Solaris Containers [TC2004] are examples of systems which support fine granularity of CPU resources and support sharing of CPUs between partitions. Physical partitioning systems such as HP Node Partitions [HP2004i] and Sun Dynamic System Domains [SM2004f] only support allocating all of the processors on a system board.

The response type for this criterion is also specified as a numerical range. The finest level of granularity supported by the system under consideration is recorded as the lower bound. If the finest level of granularity in another configuration is coarser, this is specified as the upper bound. Consider IBM Logical Partitions [IBM2004a]. Using dedicated processor partitions the finest level of CPU resource granularity is a single CPU. If shared processor partitions are used, the granularity is much finer. The comment section of the response is used to record any relevant information about configurations required to achieve these levels of granularity. Granularity is measured as a multiple of a single CPU. For systems which support very fine levels of granularity using shares, a value of 0.001 is recorded. Granularity below this level is not likely to make a significant difference.

When ranking systems the lower bound is to be used.

Response Type: Numerical range

CPU Reallocation Time Granularity (C 5.3)

The time granularity of CPU resource reallocation refers to how long it takes to migrate a CPU from one partition to another. This is straightforward to understand. If a system does not support any kind of dynamic CPU resource reallocation, this measure is not applicable.

This criterion is more relevant to physical partitioning systems than other software based approaches which provide very fine granularity.

The CPU reallocation time granularity typically depends on the configuration of the system. The best reallocation time measured is recorded as the lower bound of the numerical range for this category. Information about the configuration is recorded in the comments section. The worst CPU resource time granularity is recorded as the upper bound.

When ranking systems the lower bound is to be used.

Response Type: Numerical range

Memory Reallocation Time Granularity (C 5.4)

Similarly memory reallocation time refers to the amount of time taken to reallocate memory from one partition to another. For systems which do not support dynamic memory reallocation this criterion is not applicable.

This criterion is particularly relevant to physical partitioning systems, as these systems may require partitions to be quiesced during memory reallocation.

Similarly to the previous criterion, the lower and upper bounds represent the best and worst memory reallocation time measurements taken. Information about the configuration of the system when these values were recorded is to be included in the comments section.

When ranking systems the lower bound is to be used.

Response Type: Numerical range

5.8.6 Scalability (C 6)

Scalability is divided into two categories. The first category focuses on scaling to larger partitions, the second focuses on scaling to a larger number of partitions. Each of these categories and their importance is discussed in greater detail in the sections which follow. The two scalability categories are depicted in Figure 5.11.

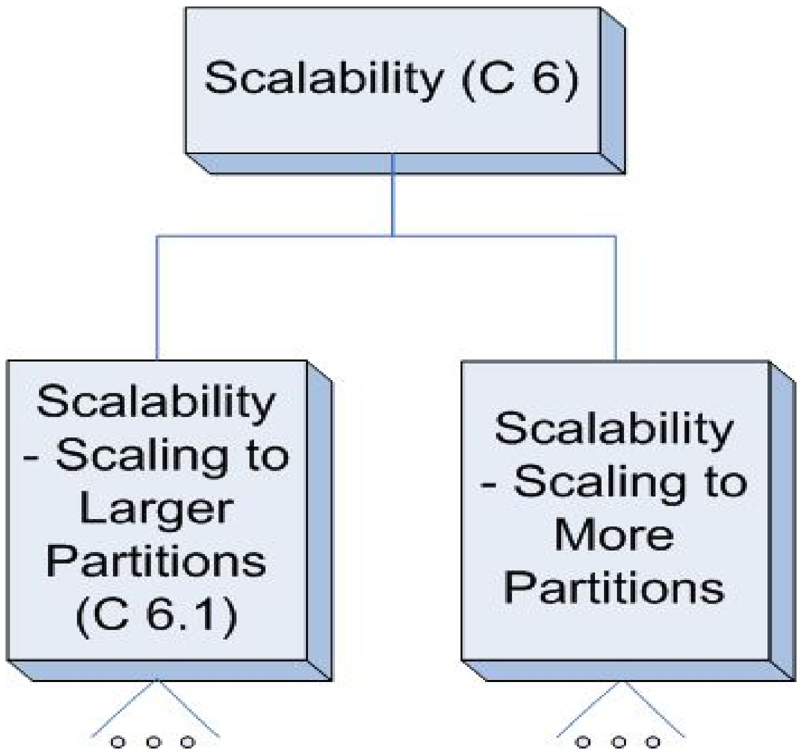

Scalability (C 6)
Scalability - Scaling to Larger Partitions (C 6.1)
Scalability - Scaling to More Partitions

Figure 5.11: Two Types of Scalability

Scalability – Scaling to Larger Partitions (C 6.1)

The first type of scalability considered is the ability to scale to larger partitions. Larger in this context refers to the quantity of resources which can be allocated to any single partition. Scaling to larger partitions is important for providing the necessary throughput needed for some applications [UVS+2004]. Larger partitions are useful for supporting different tiers of an application on the same server. This is illustrated by an example in [HP2002]. More information on tiered applications is provided in Chapter 2. Scaling to larger partitions is also important for partitioning larger servers such as the HP Superdome [HP2005d] or the Sun Fire 15k [SM2003b]. Each of these servers supports over a hundred processors.

The scalability criteria for this type of scalability are depicted in Figure 5.12 and presented in the next section.

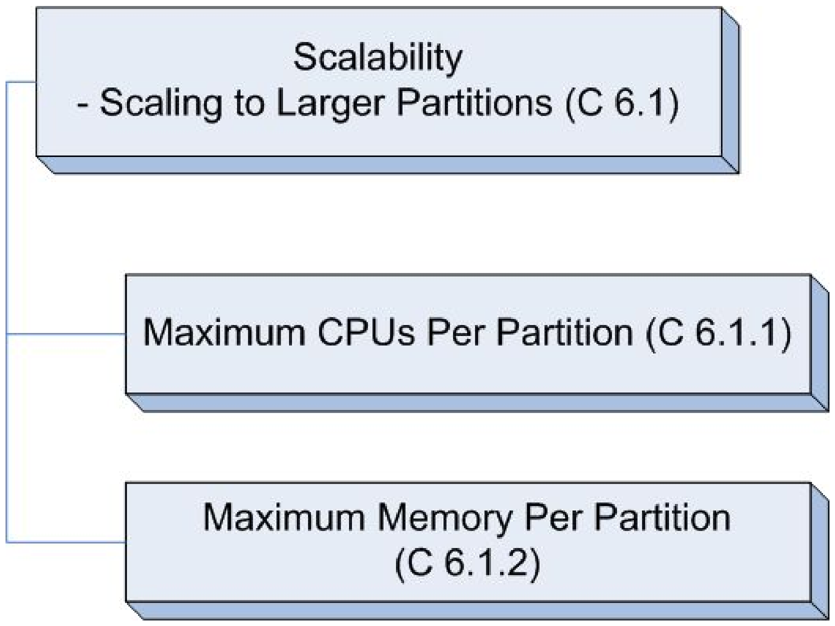

Scalability
- Scaling to Larger Partitions (C 6.1)
Maximum CPUs Per Partition (C 6.1.1)
Maximum Memory Per Partition (C 6.1.2)

Figure 5.12: Criteria for Scalability – Scaling to Larger Partitions

Maximum CPUs Per Partition (C 6.1.1)

Support for multi-CPU partitions is useful for ensuring throughput scalability which is important for server consolidation [UVS+2004]. Limits on the number of CPUs per partition are quite common and should be taken into consideration when comparing server virtualisation systems. Microsoft Virtual Server 2005 [MS2004c], which was reviewed in Chapter 4, for example, is currently limited to a maximum of one CPU per partition. Other systems such as Sun Dynamic System Domains [SM2004f] and IBM Logical Partitions [IBM2004a] are able to take full advantage of the available hardware.

The largest number of CPUs supported for a partition is recorded as a numerical range. If there is no limit documented for a system, the maximum number of CPUs in the largest server known to be compatible with the system is recorded (custom or experimental configurations excluded). Separate lower and upper bound are only recorded if a system has multiple software (e.g. operating system) dependent limits. An example of a system with this type of restriction is HP Node Partitions [HP2004c]. The term CPU is used here to refer to CPU cores. A dual core chip for example will count as two CPUs.

Response Type: Numerical range

Maximum Memory Per Partition (C 6.1.2)

Similar to the previous criterion, the maximum quantity of memory which can be assigned to a partition is now considered. With hardware such as the IBM p5-595 [IBM2005f] supporting up to 2 TB of memory there is a clear need for partitioning systems which can scale to large quantities of memory. The maximum quantity of memory supported per partition is represented as a numerical range. If there is no limit documented for a system the maximum quantity of memory supported by the largest server known to be compatible with the system is recorded (custom or experimental configurations excluded). As with the previous criterion, separate lower and upper bound values should only be recorded if a system has multiple operating system dependent limits. An example of such as system is HP Node Partitions [HP2004c].

Response Type: Numerical range

Scalability – Scaling to More Partitions (C 6.2)

The second type of scalability considered deals with factors which affect the number of partitions which can be hosted simultaneously on the same server. This type of scalability is particularly important for service providers. Scaling to many partitions or “protection domains” is therefore important [WSG2002b]. Hardware in data centres of web and application hosting providers is often shared between customers [AID2002, RMS+2000].

This will not be a critical requirement for data centres which only to host a small number of partitions per server. Some server virtualisation systems are more suitable for hosting large numbers of partitions than others. This became apparent during the investigation into the different virtualisation approaches in Chapter 3 and the study of current virtualisation systems presented in Chapter 4. It is clear that per-partition resource usage has a major impact on the number of partitions which can be hosted on a system.

The seven criteria for measuring this type of scalability are depicted in Figure 5.13 and discussed in the section which follows.

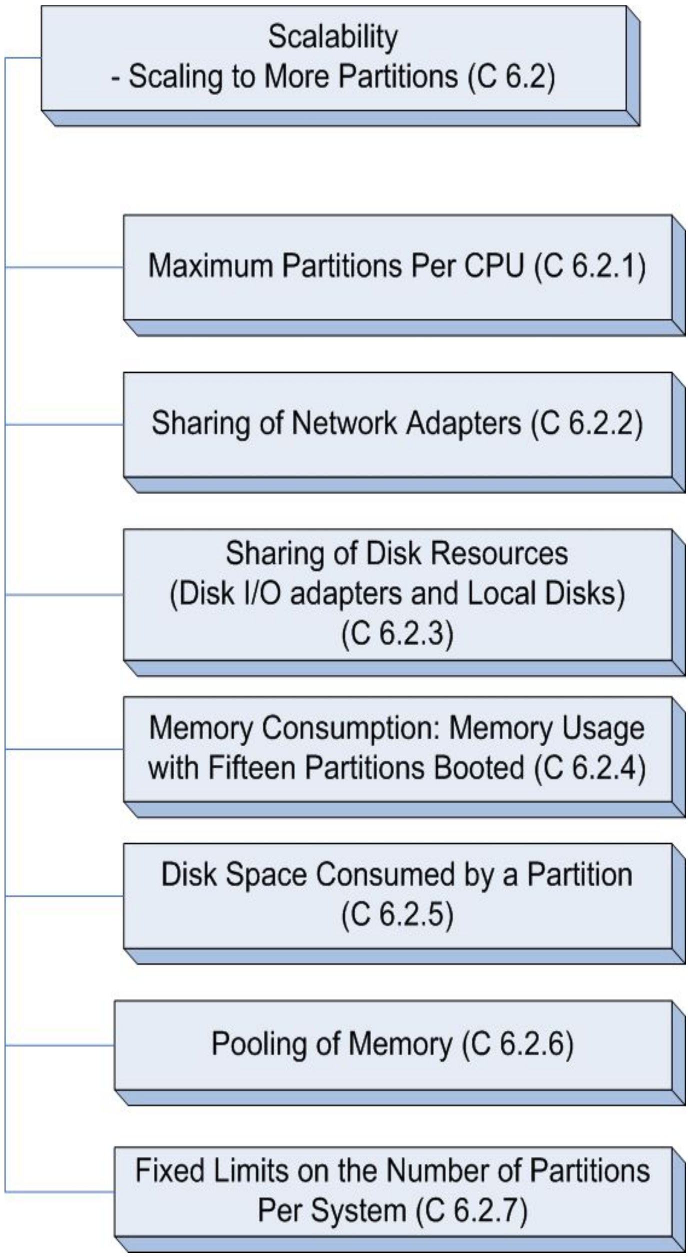

Scalability
- Scaling to More Partitions (C 6.2)
Maximum Partitions Per CPU (C 6.2.1)
Sharing of Network Adapters (C 6.2.2)
Sharing of Disk Resources (Disk I/O adapters and Local Disks) (C 6.2.3)
Memory Consumption: Memory Usage with Fifteen Partitions Booted (C 6.2.4)
Disk Space Consumed by a Partition (C 6.2.5)
Pooling of Memory (C 6.2.6)
Fixed Limits on the Number of Partitions Per System (C 6.2.7)

Figure 5.13: Criteria for Scalability – Scaling to More Partitions

Maximum Partitions Per CPU (C 6.2.1)

Restrictive limits on the number of partitions per CPU prevent some systems from scaling to a larger number of partitions. Consider the example of HP Virtual Partitions [HP2003a] which only supports one partition per CPU. For some systems, particularly physical partitioning systems such as HP Node Partitions [HP2004i] and

Sun Dynamic System Domains [SM2004f] the number of partitions per CPU may be less than one. A value greater than one indicates that CPUs can be shared by more than one partition.

This criterion is not to be confused with the CPU resource granularity criterion from the granularity category. This is because the quantity of CPU resources which can be reallocated from one partition to another is not the same as the minimum CPU resource allocation for a partition which is an important scalability consideration.

Response Type: Number

Sharing of Network Adapters (C 6.2.2)

If each partition requires a separate network adapter, this can become a factor limiting the number of partitions per system. This can be an issue when using HP Virtual Partitions because adapters cannot be shared between vPars [HP2003a].

This criterion is not to be confused with the network bandwidth isolation criterion of the resource isolation category. Whether or not a system supports network bandwidth guarantees for a shared adapter is not considered here. The only fact being considered is whether a single network adapter can be shared between multiple partitions.

Response Type: Boolean

Sharing of Disk Resources (Disk I/O Adapters and Local Disks) (C 6.2.3)

Similarly sharing of disk I/O adapters or local disks enables disks, whether local or network attached, to be utilised by multiple partitions. This reduces the likelihood of the number of adapters available being a factor limiting the number of partitions possible.

This criterion is satisfied if a single disk can or disk I/O adapter can be shared between multiple partitions.

Response Type: Boolean

Memory Consumption

Each server has a limited quantity of physical memory, therefore the memory consumed by partitions (including any overhead) is a factor which could restrict scalability. Once again this boils down to isolation versus overhead. Systems such as Xen [BDF+2003] and VMware [VMW2004a] create a separate operating system instance for each partition. Container-based systems such as Virtuozzo [Vir2005] and Solaris Containers [TC2004] share a single operating system instance between the partitions. The latter approach requires less memory and can therefore be used to create a larger number of partitions. The downside to the container approach is that it offers less isolation than separate operating system instances.

Memory consumption consists of a number of components. Some of these components such as memory consumption by containers or operating system instances contribute towards memory consumption on a per-partition basis. Others are global such as memory consumed by a virtual machine monitor or host operating system. This overhead also contributes to the amount of memory consumed. The size of this overhead usually depends on how a system is configured. This is the case with systems such as IBM Logical Partitions [IBM2005b] and VMware ESX Server [VMW2005a]. For more information on the memory overheads of these systems refer to Chapter 6. Virtual machine monitors and logical partitioning systems were discussed in Chapter 3.

One approach to reducing memory consumption is to share memory between partitions. This can take a number of forms:

One approach, which was examined in Chapter 4, is to scan for duplicate pages between partition. This approach is taken by VMware ESX Server and is described in [Wal2002]. Duplicate pages are shared between partitions. Any attempt to change the contents of one of these pages results in a copy being created (copy on write). This results in an overall reduction in memory consumption. The degree of sharing depends on the number of pages the partitions have in common.

Another approach to sharing memory is to provide partitions with a shared filesystem. Memory consumption is reduced by sharing memory mappings of files between partitions. This is the approach used for a current project, namely XenFS [Wil2005a]. IBM System/370 also used a sharing mechanism to share duplicate code pages between partitions [Gum1983].

A high degree of sharing is also inherent in container-based systems as these systems share a common operating system instance.

Based on these observations it is clear that memory consumption depends on many different factors. Initially memory consumption was broken down into multiple criteria for the framework. Each of these criteria represented a specific source of memory overhead, or approach to reducing memory consumption. This proved to be difficult given the interdependencies between the various factors affecting memory consumption. The author believed that this approach would have compromised one of the framework objectives outlined at the start of this chapter. One of the stated objectives was for the framework to be useful. As stated previously including too many criteria would limit the usefulness of the framework by making it difficult to interpret. With this in mind a much simpler solution was adopted. In order to compare the memory consumption of different systems a common measure was adopted. The total memory consumption of a system with fifteen partitions booted. This includes all of the sources of memory consumption and memory savings due to optimisations such as page sharing. This preserves the generality of the framework as specific optimisations did not have to be incorporated into criteria. Thus when new approaches to reducing memory consumption are invented there is no need to change the framework.

The memory consumption of fifteen partitions will often depend on the configuration of the system. For this reason the response type of this criterion is a numerical range. The lower and upper bounds of this range represent the minimum and maximum per-partition memory consumption values respectively. This enables both the best and worst case scenarios to be recorded for each system. Relevant information about configuration settings which resulted in these memory consumption figures is to be recorded in the comments section.

The number of fifteen was selected because multiple partitions are required to prevent the global memory consumption overhead from dominating the memory consumption measurement. It is also low enough to be within the maximum number of partition limits of all major systems.

When ranking systems using this criterion, the lower bound of the range is to be used. This was chosen with likely applications in mind. Virtual web hosting, which was identified as one of the applications of server virtualisation in Chapter 3, is the most likely application if the goal is to a large number of partitions. In such cases it is most likely that the least resource-intensive configuration will be used.

Response Type: Numerical range

Disk Space Consumed by a Partition (C 6.2.5)

Disk space consumed by each partition is another factor which can limit the number of partitions which can be hosted on a system. According to Aron *et al.* [AID2002], shared hosting providers provide service contracts which typically specify disk quotas. Thus disk space consumption is a concern when scaling to a large number of partitions. This criterion considers the disk space consumed by a newly created partition.

For systems with a large number of partitions installing many copies of an operating system consumes large quantities of disk space. VMware ESX Server [VMW2005a] and IBM Logical Partitions [IBM2004a] are examples of systems which require a separate operating system instance for each partition. Container-based systems such as Virtuozzo [Vir2005] and Solaris Containers [TC2004] consume smaller quantities of disk space per container instance. On smaller systems where disk space is more likely to be a factor these systems will scale to a larger number of virtual servers. One approach to reducing disk consumption of operating system instances is a shared filesystem such as XenFS [Wil2005a] which is currently under development.

The data type for this criterion is a numerical range. This is because the amount of disk space consumed by a partition depends on factors, such as the operating system being installed, or the number of files shared in the case of container-based systems. The disk space consumed needs to be measured for each of the common configurations. The smallest disk usage result forms the lower bound for the numerical range, and the largest the upper bound. Relevant information about configurations used to obtain these numbers needs to be included in the comments section.

When ranking systems the lower bound of the range is to be used.

Response Type: Numerical range

Pooling of Memory (C 6.2.6)

One of the benefits of virtualisation highlighted in Chapter 3 is that of improved resource utilisation due to pooling of these resources. Systems such as Microsoft Virtual Server [MS2004c] and HP Virtual Partitions [HP2003a] only provide support for allocating dedicated memory to partitions. This memory cannot be used by other partitions, even if it is not being used. Scalability is therefore restricted as memory needs to be allocated ahead of time. This results in the silo effect discussed in Chapter 2.

Pooling of memory involves all partitions sharing a global pool of memory from which processes in each partition are allocated memory. This improves resource utilisation, but will clearly come at the cost of resource isolation. Pooling of memory is supported by container-based systems. Gradual shifting of unused memory from one partition to another by workload management tools is not considered here. This was already considered in the flexibility category.

Response Type: Boolean

Fixed Limits on Number of Partitions Per System (C 6.2.7)

Some virtualisation systems have a fixed upper limit on the number of partitions per server. This includes any limit which supersedes the limits resulting from resource constraints. An example of a system with such a restriction is IBM Logical Partitions [IBM2005b] for larger pSeries servers. For more details and other examples refer to Chapter 6.

Values for this criterion are specified as numerical ranges because this limit may depend on the type of server being partitioned. Information about such dependencies is to be included in the comment section. The upper bound is to be used for ranking.

Response Type: Numerical range

5.8.7 Performance (C 7)

Virtualisation offers a number of advantages, but it is not without drawbacks. The most significant drawback is the performance overhead of many virtualisation systems. According to Belpaire and Hsu [BH1975] the performance of a virtual machine should be such that the only performance impact should be that resulting from the sharing of resources. This was identified as an "essential characteristic" of a virtual machine by this source. Efficiency was also identified by Popek and Goldberg [PG1974] and Rosenblum [Ros2004] as one of the characteristics of a virtual machine monitor.

The performance overhead of a virtualisation system depends on the implementation. The approaches to implementing virtualisation systems were discussed in Chapter 3.

Benchmark results show that virtualisation overhead has a major impact on application throughput. Some systems exhibit significantly higher performance overhead than others, as demonstrated by Barham *et al.* [BDF+2003]. Another source, Surányi *et al.* [SHH+2005], compared the relative performance of different virtualisation systems running the Apache HTTP Server [ASF2005a]. The results obtained further illustrate this point. These results are depicted in Figure 5.14.

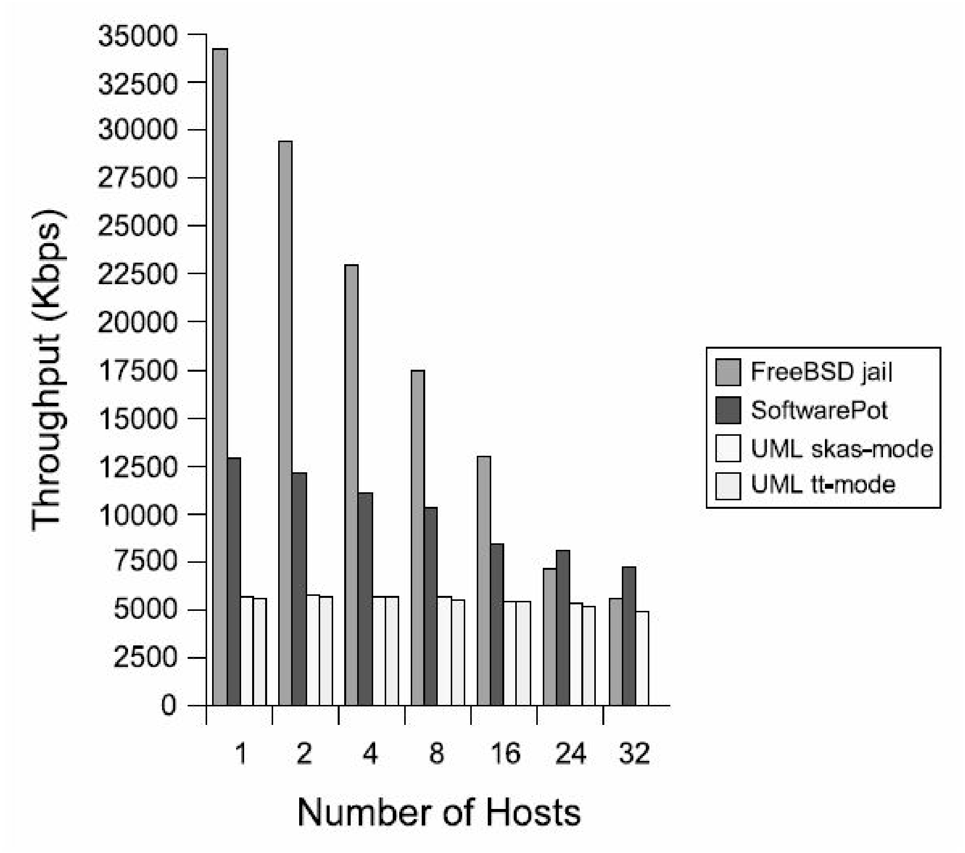


Figure 5.14: Apache Performance with Various Virtualisation Systems [SHH+2005]

Performance tests [BDF+2003, SHH+2005, MST+2005, TC2004] of virtualisation systems compare the performance of a benchmark run within a partition (virtualised) to the performance obtained when running it on a standard (non-virtualised) system. Two types of tests are used by virtualisation system authors to measure overhead and compare their systems to competing systems.

Microbenchmarks

The first approach is to use microbenchmarks to measure the performance of specific operating system facilities and operations such as system calls or forking processes. This approach is used by Barham *et al*. [BDF+2003] and Alicherry and Gopinath [AG2001].

Application Benchmarks

The second approach is to measure the performance of standard applications such as web servers or databases. This approach is taken by Barham *et al*. [BDF+2003], Surányi *et al*. [SHH+2005] and Tucker and Comay [TC2004]. It should be noted that Barham *et al*. [BDF+2003] used both approaches.

Framework Performance Criteria

It was decided that the best approach to measuring performance was to compare performance using benchmarks. Attempting to classify each system according to the system specific optimisations employed was considered impractical, due to the wide variety of systems available. This would also have resulted in a loss of generality as each new optimisation would need to be included in the framework, resulting in an unmanageably large set of criteria. A much simpler solution was selected. The efficiency of each virtualisation system can be measured in a practical manner though the use of benchmarks. By focusing on actual performance figures the framework was kept simple and easy to interpret.

Benchmark Selection

In order to ensure that the benchmarks are available to as wide an audience as possible, only benchmarks which are freely available were considered. Another challenge when selecting benchmarks is the portability of the benchmark. Virtualisation systems support a wide range of operating systems. All of the major virtualisation systems currently support Linux and/or Unix [TOG2005]. For this reason, compatibility with these operating systems was considered a minimum requirement.

A number of application and microbenchmarks were considered. It was decided that the best approach would be to focus on application benchmarks. This will be discussed in greater detail shortly.

The Apache [ASF2005a] web server benchmark and Sysbench [SB2005] on-line transaction processing (OLTP) database benchmark were selected. These benchmarks are available for a wide range of operating systems, are freely available, and open source. The Apache benchmark measures web server throughput. The Sysbench OLTP benchmark measures database transactions per second (tps). Web server and database software are representative of common data centre applications identified in Chapter 2. These applications are also relevant to common virtualisation use cases identified in Chapter 3, such as virtual web hosting and server consolidation.

The Apache benchmark was successfully used by Surányi *et al.* [SHH+2005] to measure the performance overhead of various virtualisation systems. This benchmark is also convenient as it is included with the Apache web server.

Initially the OSDB benchmark [OSD2005], which was used by Barham *et al.* [BDF+2003] to measure performance overhead, was considered. Upon investigation, the author found that this benchmark was not compatible with recent releases of the popular MySQL database [MSQ2005]. Sysbench [SB2005] was found to be a more suitable selection. It provides similar database benchmarking functionality to OSDB and is compatible with more recent releases of MySQL. This benchmark was also found to produce consistent results, and is reasonably easy to configure.

The Exclusion of Microbenchmarks

The lmbench [lmb1996] microbenchmark suite was considered initially. This benchmark was used to evaluate the performance of the Xen virtual machine monitor [BDF+2003]. One problem with microbenchmarks of this type is the accuracy of results. In one specific test by Barham *et al.* [BDF+2003] using lmbench, measurements for two identical implementations differed by thirty percent. This was ascribed to cache effects.

After over 80 test runs using this benchmark, the author decided to exclude microbenchmarks from the performance category of the framework. The latest version of this benchmark at the time of writing is lmbench 3.0-a5. The output file for each run of the operating system (OS) subset of tests was over 500 lines in length. Incorporating these tests into the framework would have effectively added hundreds of criteria. This would have made framework evaluation results very difficult to interpret, especially for practitioners, such as data centre managers. A wide variation in results of lmbench was also observed by the author for a number of the tests.

Another portable benchmark, libmicro [SM2005e], was also considered. This benchmark also includes a large number of tests (over 200). Similar to lmbench, the inclusion of hundreds of performance criteria in the framework would have made it difficult to interpret. Another concern was that the use of this benchmark may have brought the objectivity of the framework into question. This benchmark was developed by a developer at Sun Microsystems. It was used to test the performance of Solaris relative to other operating systems during the development of Solaris 10.

In addition to being simpler for practitioners to interpret, application benchmarks correspond more closely to the actual usage of these systems than microbenchmarks. The selection of two application benchmarks, which correspond to common data centre applications, ensured that the framework remained concise, yet relevant. These benchmarks will also reflect network and disk I/O performance overhead in the results.

Apache Web Server Benchmark (C 7.1)

Benchmarks are to be conducted using the Apache HTTP Server version 1.3. This application is widely used, and compatible with many operating systems. The performance of Apache on a virtualised system is to be compared with the performance on a non-virtualised system.

The Apache HTTP Server daemon is to be started on the test system. A 10 kilobyte web page is to be stored on this server. Any unnecessary system services are to be disabled. Default settings are to be used for all operating system tuning parameters and Apache settings. Another separate system is needed to act as the client for this test. This system is to be connected directly to the test system. The client system is to launch the Apache benchmark tool with the following options:

```
./ab -n 100000 -c4 http://IP-Address/tenK.html
```

This simulates four clients sending 100000 requests. These settings were based on those used by Surányi *et al*. [SHH+2005].

The client system is to run this test six times, with a thirty second gap after each. The transfer rate, which is clearly indicated in the test output, is to be recorded in each case. The result of the first test is to be discarded as a warm up run. The mean of the remaining five results is to be recorded, along with the standard deviation. This information, along with the system configuration is to be recorded in the comments section.

After reconfiguring the server system to run as a virtualised system, both the server and client test systems must be rebooted. The test process is then repeated for the virtualised case.

Once the throughput of the virtualised and non-virtualised cases have been measured these figures are to be recorded in kilo***bits*** per second. The throughput of the virtualised case is to be reported as a percentage of the performance of the non-virtualised case. If the performance overhead is low, this figure should approach 100%.

The highest percentage recorded for a given virtualisation system represents the upper bound for the numerical range. The lowest percentage recorded forms the lower bound.

Response Type: Numerical range

Sysbench OLTP Database Benchmark (C 7.2)

Unlike the Apache benchmark, Sysbench is to be run on the same system as the database server. MySQL 4.0 is to be used as the database server. The on-line transaction processing (OLTP) test is to be run using the default settings. Unnecessary system services are to be disabled and tuning parameters should not be adjusted. As with the previous criterion the benchmark is to be run six times, with a thirty second delay after each run. The first run is discarded as a warm up. The number of transactions per second (tps) is to be recorded. The mean and standard deviation are then calculated. Once the non-virtualised configuration has been tested, the virtualised configuration must be tested. The system must be rebooted before testing the virtualisation configuration.

Once the tests are completed, the mean number of transactions per second achieved with the virtualised configuration is to be expressed as a percentage of the corresponding value for the non-virtualised configuration. Information about the system configuration and standard deviation is to be recorded in the comments section for this criterion.

As with the previous criterion, the highest percentage recorded for a given virtualisation system represents the upper bound for the numerical range. The lowest percentage recorded forms the lower bound.

Response Type: Numerical range

Other Considerations

The purpose of the performance criteria is not to measure the overall performance of server equipment, but to measure the performance impact of a virtualised configuration relative to a non virtualised configuration. To ensure transparency, performance evaluation results will need to be classified according the server equipment used to conduct the evaluation.

The performance section of the framework does not apply to physical partitioning systems. There is no performance overhead for physical partitioning systems, rendering any measurement pointless. If a system no longer provides a non-virtual mode of operation, a relative performance comparison is not possible.

The Exclusion of Cost as a Framework Criterion

The cost of commercial products was not included as one of the framework criteria. There are a number of reasons for this. The framework is only concerned with criteria which are based on technical merit. Pricing information is also volatile and subject to external economic factors which have little or no relevance to the problem at hand. Furthermore, pricing information for most high-end servers is not readily available.

5.9 Interpreting Results and Applying the Framework

General Interpretation and Identifying Strengths and Weaknesses

Identifying the relative strengths and weaknesses of virtualisation systems was identified as one of the applications of the framework. This application is demonstrated in Chapter 6.

In this section a general discussion on how to interpret the results of evaluations using the framework criteria is presented. This is relevant to identifying the relative strengths and weaknesses of different systems.

Each of the non-leaf nodes of the framework represents a category. These categories all represent characteristics which were identified as desirable for server virtualisation systems in data centres. For each category the systems which have been evaluated are ranked. This ranking system was introduced earlier in this chapter. A category ranking is intended as a quick reference, akin to an approximation. This can be used to identify general strengths and weaknesses. Inspection of the lower level evaluation data in the leaf nodes is recommended for a more detailed analysis. This is recommended because the effect of a category ranking is to summarise lower level information resulting in a loss of detail. To demonstrate this, consider a hypothetical category consisting of six boolean criteria. Two systems are evaluated and each satisfies five of the six criteria. Even though each system satisfies the same number of criteria, these are not necessarily the same five criteria. Inspection of the evaluation data is therefore an important part of a comparison.

The leaf nodes of the framework represent lower level criteria which correspond to specific characteristics or features. These nodes can be used to determine the extent to which a given system exhibits the higher-level characteristic represented by a parent node.

Ranking is also performed for each of the low-level criteria. These rankings are based directly on the low-level data. These rankings are an accurate reflection and not subject to interpretation the way category rankings spanning multiple criteria are. These low-level evaluations can be used to pinpoint specific weaknesses compared to other systems. An inspection of the low-level information is also recommended to supplement leaf node rankings.

No attempt is made to determine an overall ranking spanning multiple categories. The reasons for this were discussed earlier in this chapter in the section on the rankings system. The most suitable system for a given user is requirements dependent. With this in mind we return to one of the applications of the framework proposed earlier in this chapter, namely the use of the framework to define requirements.

5.9.1 Virtualisation Requirements Filters

In order to define requirements for a given user or in the design phase of a new virtualisation system the concept of a virtualisation requirements filter will now be introduced. The application of requirements filters, to provide users with a means of specifying requirements in order to select a suitable virtualisation system, will be discussed first. Users in this context are decision-makers such as data centre managers. Chapter 7 provides an example of a virtualisation requirements filter which was constructed to facilitate the selection of a virtualisation system for a small data centre.

Facilitating System Selection

The selection of a suitable virtualisation system given a set of requirements corresponds to the classification problem from the domain of expert systems. The following is a definition of the classification problem:

"Given a specimen which belongs to one of the object types (classes), establish the property values applicable to the properties of the specimen in order to determine its class or most likely classes" - de Kock [GdK2004]

In the context of server virtualisation, the objective is to identify a set of virtualisation systems with properties which satisfy a set of requirements (properties). The concept of a virtualisation requirements filter which is proposed here is based on production rules for expert systems. Production rules are discussed by de Kock [GdK2004].

Filter Production Rules

A virtualisation requirements filter consists of a set of rules defined by a user. The rules specify a set of requirements based on the framework criteria. It should be noted that in practice users would specify rules indirectly with the aid of a GUI-oriented decision support tool. This would shield users from theoretical details such as the syntax of rules. This discussion presents the theory which could form the foundation of such a tool. The development of a decision support tool is beyond the scope of this research.

Using these rules the requirements filter "filters" the list of available virtualisation systems contained in a candidate set to obtain a set of recommended systems. This is the reason the name "virtualisation requirements filter" was selected. As the rules are applied the candidate set is narrowed down. The only exception is the first rule. The first rule (R1) does *not* narrow this set. It populates the candidate set with systems for which evaluations are available. An example of this type of rule is as follows:

R1: add {Sun Dynamic System Domains, HP Node Partitions, Fujitsu Physical Partitions, Fujitsu Extended Partitions, HP Integrity Virtual Machines, HP Virtual Partitions, Solaris Containers, IBM Logical Partitions (pSeries), Microsoft Virtual Server, VMware ESX Server}

Obviously the items in this set will depend on the number of evaluations available. Subsequent rules can only remove items from the candidate set.

Rule Priorities

As noted previously the importance of different criteria is dependent on a user's individual requirements. For this reason priority needs to play a role when defining requirements. Rules are specified by users in descending order of importance. These rules are also applied in this order. This results in rule R2 being the most important rule. When a rule is applied all of the items remaining in the candidate set are evaluated before proceeding to the next rule.

Without prioritisation a filter would provide a sub-optimal recommendation. Consider the example of a user who specifies a number of requirements in the form of a requirements filter. This user could specify an unimportant requirement as one of the first rules. If this rule were applied first, systems which meet more important requirements could be eliminated prematurely.

Users may specify requirements which are too strict, resulting in all of the candidate systems being eliminated. In such cases the recommended systems would be those which were eliminated last. These systems may not have met every requirement, but would be closest matches. Users may also wish to consider systems which were eliminated if the most suitable systems are too expensive.

Filter Production Rules Defined

The rules for a filter will be referred to as filter production rules. Filter production rules are defined similarly to production rules. For a definition of production rules refer to de Kock [GdK2004]. The definition in Table 5.2 is adapted from this source. Key differences which distinguish *filter* production rules include:

- Only one conclusion for each rule
- Remove is the only valid action
- Additional predicates for sets are added (contains, notcontains)
- Additional predicates for numerical ranges added (rangecontains, notrangecontains, belowrange, aboverange, notbelowrange, notaboverange)

The first rule (R1) of a requirements filter is an exception and is always of the form:

R1: add {item(1), item(2), ..., item(n)}

Filter production rules are defined in Backus Normal Form (BNF) in Table 5.2.

Table 5.2 Definition of a Filter Production Rule (in Backus Normal Form)

<filter production rule>	::=	**if** <antecedent> **then** <conclusion> **fi**
<antecedent>	::=	<condition> {**and** <condition>}*
<condition>	::=	<literal> {**or** <literal>}*
<literal>	::=	<predicate> (current.<attribute>, <value>)
<conclusion>	::=	remove (current)
<predicate>	::=	same | notsame | greater | less | contains | notcontains | empty | notempty | rangecontains | notrangecontains | belowrange | aboverange | notbelowrange | notaboverange

Current represents the item from the candidate set currently being considered.

The additional predicate types are defined as follows:

List Predicates

As the list data type does not contain duplicate values, any list L can be represented by a corresponding set S of elements. If L contains a single element, “none”, then S is the empty set. If not, each element in L has a corresponding element in S. The following predicates are defined in terms of S. Each attribute is assumed to have a list data type.

empty : empty (current.attribute) is **true** if $S = \phi$, **false** otherwise

notempty :notempty (current.attribute) is **true** if $S \neq \phi$, **false** otherwise

contains : contains (current.attribute, x) is **true** if $x \in S$, **false** otherwise

notcontains : notcontains (current.attribute, x) is **true** if $x \notin S$, **false** otherwise

Numerical Range Predicates

A numerical range is a set S of real numbers with an upper bound z and a lower bound y such that:

Any value $x \in S$ if $x \in \mathbb{R}$ and $y \leq x \leq z$ where $y, z \in \mathbb{R}$ and $y \leq z$

The following predicates are defined in terms of S for numerical ranges with a lower bound of y and an upper bound of z. Each attribute is assumed to have a numerical range data type and $x \in \mathbb{R}$.

rangecontains : rangecontains (current.attribute, x) is **true** if $y \leq x \leq z$, **false** otherwise

notrangecontains : notrangecontains (current.attribute, x) is **true** if $x < y$
or $x > z$, **false** otherwise

This is equivalent to: belowrange(current.attribute, x) or aboverange (current.attribute, x)

belowrange : belowrange (current.attribute, x) is **true** if $x < y$, **false** otherwise

aboverange : aboverange (current.attribute, x) is **true** if $x > z$, **false** otherwise

notbelowrange : notbelowrange (current.attribute, x) is **true** if $x \geq y$, **false** otherwise

notaboverange : notaboverange (current.attribute, x) is **true** if x ≤ z, **false** otherwise

Rule Examples

The following are examples of filter production rules:

if aboverange(current.“CPU Resource (Re)Allocation Quantity”, “0.5”) **then** remove (current) **fi**

if notsame(current.“filesystem isolation”, “Yes”) **then** remove (current) **fi**

if notcontains(current.“operating system compatibility”, “Linux”) **then** remove (current) **fi**

if greater (current.“Memory resource (re)allocation quantity”, “16 MB”) **and** notcontains (current.“Memory Resource Isolation”, “Capping”) **then**
remove (current) **fi**

It is recommended, for readability and consistency, that requirements be specified as positives. Thus in order to eliminate unsuitable systems predicates are typically specified in the negative.

Guiding Requirements Definition for a New System

Another potential application could be to use a filter to specify requirements for a new virtualisation system during the design phase. These requirements could be identified with the aid of the framework by examining evaluations of current systems.

5.10 Generality and Extensibility of the Framework

The framework was designed to be applicable to a wide range of virtualisation systems. It is not restricted to any particular set of implementation approaches. The high-level framework criteria (categories) are based on general characteristics such as compatibility and granularity. This makes it possible to extend the framework to include additional low-level criteria in future.

5.11 Conclusion

The goal of the framework presented in this chapter was to provide an objective set of criteria for comparing server virtualisation systems. The usefulness and applicability of this framework are demonstrated in Chapters 6 and 7. Chapter 6 documents the evaluation results of ten virtualisation systems using the framework. These systems are also ranked in each of the framework categories using the ranking system which was presented in this chapter. This ranking system is kept reasonably simple yet accurate despite, the complex nature of the information. The accuracy of the category ranking system is discussed further in Chapter 6. The evaluations in Chapter 6 are based on literature. Chapter 7 provides a case study of a complete evaluation incorporating practical aspects and experimentation. The use of requirements filters to assist in the selection of a suitable virtualisation system is also demonstrated in Chapter 7

Designing this framework was not a straightforward task. Achieving a clean separation between the various framework categories proved to be a considerable challenge. The resulting tree structure represents the clean separation achieved.

The need to balance the requirements of thoroughness and simplicity was identified at an early stage of the framework's design. To realise this goal the framework focuses on a manageable set of relevant criteria. These criteria were selected based on characteristics which were identified as the most useful for virtualisation systems in data centres.

The filtering mechanism proposed in this chapter forms the theoretical foundation for a decision support system based on the framework criteria. Although the implementation of such a system is beyond the scope of this research, it does provide an avenue for future work.

One of the achievements of the framework was to provide a structure onto which relevant information about virtualisation systems can be mapped. This is particularly valuable given the fact that such information is often unstructured and dispersed among multiple sources.

Chapter 6: Evaluation of Existing Systems Using the Framework

6.1 Introduction

The design of the framework was presented in Chapter 5. In order to realise the benefits of this framework, such as facilitating decision-making and identifying the relative strengths and weaknesses of systems, existing virtualisation systems need to be evaluated. The use of the framework to evaluate a number of current virtualisation systems is demonstrated in this chapter. These evaluations serve a number of purposes:

- Demonstrate the practicality of the framework
- Populate the framework with useful information
- Provide an example for others to follow when evaluating systems using the framework
- Demonstrate how the framework can be used to highlight the strengths and weaknesses of each system
- Demonstrate the ranking system described in Chapter 5

These evaluations are based on a literature study. The systems selected for evaluation were evaluated against all of the framework criteria which could be evaluated from literature. This was for practical reasons. A few of the framework criteria, such as those requiring performance measurements, need to be evaluated experimentally. Evaluating all of these systems using experimentation would require purchasing and deploying an example of each system. Budgetary constraints alone made this infeasible. An evaluation based on literature is valid because this literature included detailed system manuals and white papers, which contain detailed technical information about these systems.

An example of a complete evaluation, including practical aspects, is presented in Chapter 7. Solaris Containers is the subject of this evaluation. This system was selected with the aid of the framework as the most suitable system for use in the NMMU Telkom CoE data centre. The system selection process is also presented in Chapter 7.

Ten major virtualisation systems have been evaluated using the framework. The selected systems are (in no particular order) HP Node Partitions, HP Virtual Partitions, Fujitsu Extended Partitions, Fujitsu Physical Partitions, HP Integrity Virtual Machines, IBM Logical Partitions, Sun Dynamic System Domains, Solaris Containers, VMware ESX Server 2.5 and Microsoft Virtual Server 2005. Seven of these systems were discussed in Chapter 4. In order to test the practicality of evaluating systems using the framework, three systems are evaluated which were not considered during the design phase of the framework. These systems are Fujitsu Extended Partitions, Fujitsu Physical Partitions and HP Integrity Virtual Machines.

This chapter consists of three main sections:

- Comparative evaluations using the framework
- Strengths and weaknesses analysis
- Evaluation feasibility

6.2 Evaluations

In this section comparative evaluations of ten current virtualisation systems are presented. These evaluations are based on information which is current on 22 November 2005. The evaluation data is presented in tabular form for each of the low-level criteria. A general discussion is provided for each of the categories. The ranking system which was introduced in Chapter 5 was used to rank the systems. For brevity the category rankings will be presented in the form of a bar chart for each category. The rankings for each of the low-level criteria are included in the tables alongside the evaluation data.

Evaluating current virtualisation systems using the framework also serves to test the applicability of the criteria selected in the previous chapter. As stated in the introduction, this evaluation is based on available literature. Some of the framework criteria are not applied in this section, as these criteria require having each of the systems to test. These criteria are clearly marked as such in the sections which follow.

The merits of the framework criteria selected were covered in Chapter 5. The commentary on the evaluations which follow focuses on how the different systems fare against the framework criteria. This serves to critically compare the various systems and their capabilities according to the framework criteria.

Compatibility (C 1)

The compatibility evaluation results are presented in tabular form in Tables 6.1 to 6.5.

It is clear from Table 6.1 that the majority of the virtualisation systems evaluated are restricted to a single instruction set architecture (ISA). This restricts the type of hardware on which a system can be used. The exceptions are Solaris Containers, HP Node Partitions and HP Virtual Partitions. These systems are ranked above the others for this criterion. Three of the systems evaluated support the common x86 ISA.

Seven of the ten systems evaluated are restricted to a single operating system. Three systems, namely VMware ESX Server, HP Node Partitions and IBM Logical Partitions, support multiple operating systems. As a result, these systems are ranked above the others for this criterion. The operating systems supported by these systems are listed in Tables 6.2a and b. These systems can all host multiple operating system instances simultaneously on the same server. Sun Dynamic System Domains, the Fujitsu partitioning systems and Microsoft Virtual Server 2005 support different versions of the same operating system simultaneously. Refer to Tables 6.5a and b for more details.

Interestingly nPartitions, ESX Server and LPARs support multiple operating systems and a single ISA (IA64, x86 and POWER respectively), whereas Solaris Containers support a single operating system (Solaris) and two ISAs (x86 and SPARC).

Driver support is unlikely to be an issue with most of the systems evaluated as most are designed for and supported on vendor specific hardware. The only case where driver support is significantly different between the virtualised and non-virtualised cases is with VMware ESX Server. This is reflected in Tables 6.3a and b, where this system is ranked below the others.

Solaris Containers appears to be the only system evaluated which may be incompatible with some existing applications. While the vast majority of existing applications are compatible with Solaris Containers, there are a number of restrictions. Containers cannot be used as NFS servers for example, nor can a user of a partition load a kernel module. These and other similar restrictions are unlikely to be an issue for most applications. All of the systems evaluated satisfy this criterion, as per the requirements stipulated in Chapter 5. For this reason all of the systems in Table 6.4 are ranked in first place for this particular criterion.

Table 6.1: Instruction Set Architectures Supported (C 1.1)

Virtualisation System	*Rank*	*Instruction Set Architectures (ISAs) Supported*
HP Node Partitions	1	PA-RISC, IA64 (Itanium)
HP Virtual Partitions	1	PA-RISC, IA64 (Itanium)
Solaris 10 Containers	1	SPARC, x86
Fujitsu Extended Partitioning	2	SPARC
Fujitsu Physical Partitioning	2	SPARC
HP Integrity Virtual Machines	2	IA64 (Itanium)
IBM Logical Partitions	2	POWER
Microsoft Virtual Server 2005	2	x86
Sun Dynamic System Domains	2	SPARC
VMware ESX Server 2.5	2	x86

Table 6.2a: Operating System Compatibility (C 1.2)

Virtualisation System	*Rank*	*Operating System Compatibility*
VMware ESX Server 2.5	1	Windows, Linux, NetWare, Solaris, FreeBSD.
		Comment: Refer to the guest OS installation guide for more information [VMW2005c]
HP Node Partitions	2	HP-UX, Windows, Linux, OpenVMS **Comment:** Windows, OpenVMS and Linux are only supported on Itanium [HP2005i]
IBM Logical Partitions	3	AIX, Linux, i5/OS
Fujitsu Extended Partitioning	4	Solaris
		Comment: Requires Solaris 8 or higher [FS2005a]
Fujitsu Physical Partitioning	4	Solaris
		Comment: Requires Solaris 8 or higher [FS2005a]

(continued...)

Table 6.2b: Operating System Compatibility (C 1.2)

Virtualisation System	*Rank*	*Operating System Compatibility*
HP Integrity Virtual Machines	4	HP-UX
		Comment: Requires HP-UX 11i v2 May 2005 or later [HP2005g]
HP Virtual Partitions	4	HP-UX
		Comment: HP-UX 11i v1 or later for PA-RISC. HP-UX 11i v2 or later required for Itanium [HP2005h].
Microsoft Virtual Server 2005	4	Windows
		Comment: Other x86 operating systems which are not officially supported are compatible
Solaris 10 Containers	4	Solaris
		Comment: Only Solaris 10 is currently supported
Sun Dynamic System Domains	4	Solaris
		Comment: Requires Solaris 8 or higher [SM2004f]

Table 6.3a: Device Compatibility (C 1.3)

Virtualisation System	*Rank*	*Device Compatibility*
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes
HP Integrity Virtual Machines	1	Yes
		Comment: Requirement satisfied trivially as only a selected set of IBM servers needs to be supported
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
		Comment: Lack of support for graphics cards or audio is not relevant to server applications.
IBM Logical Partitions	1	Yes
		Comment: Requirement satisfied trivially as only a selected set of IBM servers needs to be supported

(continued...)

Table 6.3b: Device Compatibility (C 1.3)

Virtualisation System	***Rank***	***Device Compatibility***
Microsoft Virtual Server 2005	1	Yes
		Comment: Uses emulated device drivers for virtual machines which interact with devices via device drivers of the host. This provides support for many standard devices. Sound cards are not currently supported. [MS2004c]
Solaris 10 Containers	1	Yes
		Comment: Although Solaris compatibility is limited to a hardware compatibility list, there is no difference between the device support in the virtualised and non-virtualised cases.
Sun Dynamic System Domains	1	Yes
VMware ESX Server 2.5	2	No
		Comment: Only a certified set of devices are supported by the ESX Server VMM. This restricts compatibility compared to the non-virtualised case. Supported hardware is listed in a compatibility guide [VMW2005b].

Table 6.4: Compatibility with Existing Applications (C 1.4)

Virtualisation System	*Rank*	*Compatibility with Existing Applications*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
		Comment: Most applications are compatible, but there are exceptions. Examples include kernel modules and applications which require a partition to act as an NFS server.
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Table 6.5a: Concurrent Operating Systems Compatibility (C 1.5)

Virtualisation System	***Rank***	***Concurrent Operating Systems Compatibility***
HP Node Partitions	1	Different OS [HP2004c], OS version [HP2004b], OS patch level
IBM Logical Partitions	1	Different OS, OS version, OS patch level
VMware ESX Server 2.5	1	Different OS, OS version, OS patch level
Fujitsu Extended Partitioning	2	OS version, OS patch level
		Comment: (only Solaris is supported) [FS2005a]
Fujitsu Physical Partitioning	2	OS version, OS patch level
		Comment: (only Solaris is supported) [FS2005a]
HP Integrity Virtual Machines	2	OS version, OS patch level
		Comment: Different versions and patch levels of HP-UX [HP2005g]
Microsoft Virtual Server 2005	2	OS version, OS patch level
		Comment: Only Windows is officially supported.

(continued...)

Table 6.5b: Concurrent Operating Systems Compatibility (C 1.5)

Virtualisation System	*Rank*	*Concurrent Operating Systems Compatibility*
Sun Dynamic System Domains	2	OS version, OS patch level
		Comment: Different versions of Solaris supported [SM2003b].
HP Virtual Partitions	3	OS patch level
		Comment: Only HP-UX is supported [HP2004c]. All HP-UX instances must be the same version [HP2005h] within the same node (server or nPartition). Supports different patch levels of HP-UX 11i [HP2004k]
Solaris 10 Containers	4	None
		Comment: A single OS instance is shared between all containers.

Category Ranking

The overall ranking results for compatibility are depicted in Figure 6.1. These results are based on the ranking system presented in Chapter 5. Unlike individual criteria, where smaller values are better, higher category ranking values are better. This is because category ranking values represent the average number of points scored by a system in a category. Rankings for a specific low-level criterion represent positions.

As discussed in Chapter 5, the category ranking system is intended as a quick reference, or approximation, to indicate the relative strengths and weaknesses of systems within a category. In Figure 6.1 HP Node Partitions fared the best, followed by IBM Logical Partitions and VMware ESX server. Node Partitions support two ISAs and four operating systems whereas Logical Partitions support a single ISA and three operating systems. VMware ESX Server supports more operating systems than any of the others, but only supports a single ISA (x86). This, together with limited device driver support, resulted in this system being ranked in third place. Virtual Partitions and Solaris Containers were boosted by the fact that they each support two ISAs. The remaining systems all support only a single operating system (although multiple versions) on a single ISA. It is debatable whether Solaris Containers should have been ranked above systems which support multiple versions of the same operating system. The reason for this was that it supports two ISAs, boosting its score. As emphasised in Chapter 5, manual inspection of the low-level evaluation data is always recommended.

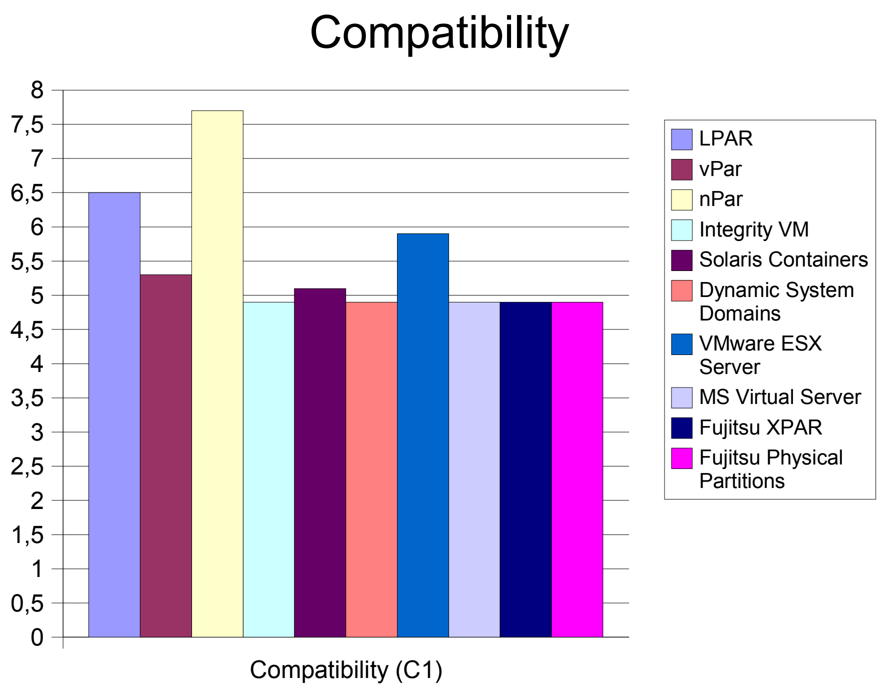


Figure 6.1: Ranking Results for the Compatibility Category

Isolation (C 2)

The isolation features of each system are evaluated according to three groupings. These are discussed separately in the sections which follow.

Fault Isolation (C 2.1)

The fault isolation evaluation results are presented in Table 6.6 and Table 6.7.

Fault isolation is a clear strength of physical partitioning systems such as Sun Dynamic System Domains, HP Node Partitions and the Fujitsu partitioning systems. This provides isolation from hardware failures. These systems are ranked in first place for this criterion. This is reflected in Table 6.6.

Software isolation is provided by nearly all of the systems evaluated. These systems have a separate operating system instance for each partition. This provides isolation from operating system failures in other partitions. The only exception is Solaris Containers. This system shares a single operating system instance between all partitions. For this reason this system is ranked behind all of the others for this criterion. This information is summarised in Table 6.7.

Table 6.6: Physical Isolation Between Partitions (C 2.1.1)

Virtualisation System	*Rank*	*Physical Isolation Between Partitions*
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes
HP Node Partitions	1	Yes
Sun Dynamic System Domains	1	Yes
HP Integrity Virtual Machines	2	No
HP Virtual Partitions	2	No
IBM Logical Partitions	2	No
Microsoft Virtual Server 2005	2	No
Solaris 10 Containers	2	No
VMware ESX Server 2.5	2	No

Table 6.7: Separate Operating System Kernel for Each Partition (C 2.1.2)

Virtualisation System	***Rank***	***Separate Operating System Kernel for Each Partition***
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes
HP Integrity Virtual Machines	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
IBM Logical Partitions	1	Yes
Microsoft Virtual Server 2005	1	Yes
Sun Dynamic System Domains	1	Yes
VMware ESX Server 2.5	1	Yes
Solaris 10 Containers	2	No

Category Ranking

The ranking results for the fault isolation category are depicted in Figure 6.2. The superior isolation provided by the physical partitioning systems is reflected in the ranking results. Solaris Containers received the lowest score. This is because it shares a single operating system instance between partitions.

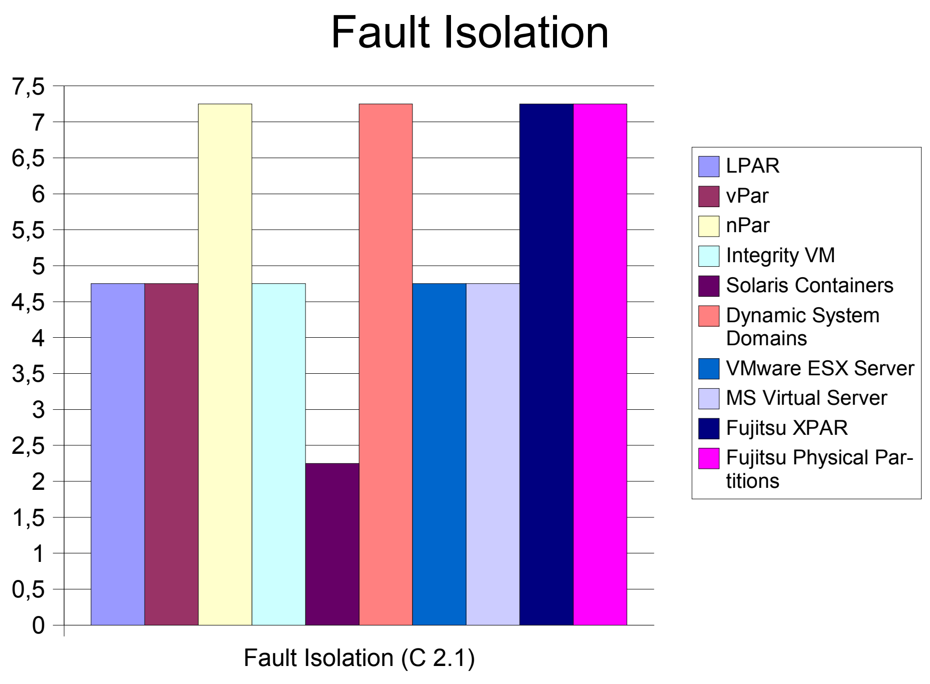


Figure 6.2: Ranking Results for the Fault Isolation Category

Resource Isolation (C 2.2)

The resource isolation criteria focus on four resources, namely CPU, memory, disk and network bandwidth resources. The evaluations are presented in Tables 6.8 to 6.12.

Seven of the systems evaluated provide support for dedicating CPUs to partitions in at least one configuration. These systems are ranked in first place. Some systems can mix dedicated and scheduled CPUs. IBM LPARs, for example, can host both dedicated and shared processor partitions simultaneously on the same system. Similarly Solaris Containers supports both shares based CPU allocation and processor sets.

Scheduling of shared CPU resources is supported by half of the systems evaluated. Only the physical partitioning systems do not support this feature. Systems which only provide this type of isolation are ranked behind the others. For more detailed information refer to Table 6.8.

Nearly all of the systems evaluated can allocate a guaranteed quantity of memory to each partition. Solaris Containers is the only system which does not support this. Only memory usage capping is supported currently. As a result, Solaris Containers is ranked last in Table 6.9b.

All of the systems under consideration support dedicating hard disks to partitions in one form or another. Only VMware ESX Server supports disk bandwidth guarantees for shared disks. Systems which support sharing of disks without any bandwidth guarantees are ranked below the other systems. This information is presented in Table 6.10.

Disk space quotas can be enforced using any of the ten systems evaluated. This is typically achieved using disk partitions or dedicated disks. This resulted in all of the systems being ranked in first place in Table 6.11.

Support for dedicating network adapters to partitions is also widespread. Only VMware ESX Server and Solaris Containers support per-partition scheduled network bandwidth guarantees. The three systems which cannot provide network bandwidth guarantees, despite supporting sharing of such adapters, are ranked last. For more details refer to Table 6.12.

Overall there was very little support for capping usage of shared resources in all the systems evaluated.

Table 6.8: CPU Resource Isolation (C 2.2.1)

Virtualisation System	*Rank*	*CPU Resource Isolation*
Fujitsu Extended Partitioning	1	Dedicated
Fujitsu Physical Partitioning	1	Dedicated
HP Node Partitions	1	Dedicated
HP Virtual Partitions	1	Dedicated
IBM Logical Partitions	1	Dedicated, scheduled, scheduled capping
		Comment: Dedicated processor partitions provide dedicated CPU support. Shared processor partitions provide scheduler-based isolation. Scheduled capping is also supported [IBM2004f].
Solaris 10 Containers	1	Dedicated, scheduled
		Comment: Dedicated CPU functionality provided by processor sets. A fair share scheduler provides scheduled isolation.
Sun Dynamic System Domains	1	Dedicated
HP Integrity Virtual Machines	2	Scheduled
		Comment: Only virtual CPUs are supported [HP2005g]. More than one virtual CPU can share a single physical CPU, thus only scheduled isolation is provided.
Microsoft Virtual Server 2005	2	Scheduled [MS2004c], scheduled capping
VMware ESX Server 2.5	2	Scheduled
		Comment: More than one virtual CPU can be scheduled to run on the same physical CPU (inferred from [IBM2005c]).

Table 6.9a: Memory Resource Isolation (C 2.2.2)

Virtualisation System	*Rank*	*Memory Resource Isolation*
Fujitsu Extended Partitioning	1	Guaranteed allocation
		Comment: Memory dedicated to each partition [FS2005a]
Fujitsu Physical Partitioning	1	Guaranteed allocation
		Comment: Memory dedicated to each partition [FS2005a]
HP Integrity Virtual Machines	1	Guaranteed allocation
		Comment: Memory dedicated to each partition [HP2005g]
HP Node Partitions	1	Guaranteed allocation
		Comment: Memory dedicated to each partition
HP Virtual Partitions	1	Guaranteed allocation
		Comment: Memory dedicated to each partition

(continued...)

Table 6.9b: Memory Resource Isolation (C 2.2.2)

Virtualisation System	*Rank*	*Memory Resource Isolation*
IBM Logical Partitions	1	Guaranteed allocation
		Comment: Memory dedicated to each partition
Microsoft Virtual Server 2005	1	Guaranteed allocation
		Comment: A fixed quantity of memory is dedicated to each partition and cannot be over committed [MS2004c]
Sun Dynamic System Domains	1	Guaranteed allocation
		Comment: Memory dedicated to each partition
VMware ESX Server 2.5	1	Guaranteed allocation
Solaris 10 Containers	2	Capping
		Comment: Only capping of physical memory usage is supported. This is enforced asynchronously using a daemon [SM2005d]

Table 6.10: Disk Bandwidth Isolation (C 2.2.3)

Virtualisation System	*Rank*	*Disk Bandwidth Isolation*
Fujitsu Extended Partitioning	1	Dedicated [FS2005a]
Fujitsu Physical Partitioning	1	Dedicated [FS2005a]
HP Node Partitions	1	Dedicated
HP Virtual Partitions	1	Dedicated
Sun Dynamic System Domains	1	Dedicated
VMware ESX Server 2.5	1	Dedicated, scheduled
		Comment: A shares based approach is used for scheduling [VMW2005a].
HP Integrity Virtual Machines	2	Dedicated, unscheduled shared [HP2005g]
IBM Logical Partitions	2	Dedicated, unscheduled shared
Microsoft Virtual Server 2005	2	Dedicated, unscheduled shared
Solaris 10 Containers	2	Dedicated, unscheduled shared
		Comment: Disks can be dedicated by ensuring that only one zone is installed on the disk to be dedicated [SM2005d]. No disk scheduling provided.

Table 6.11: Disk Space Usage Limits (C 2.2.4)

Virtualisation System	*Rank*	*Disk Space Usage Limits*
HP Integrity Virtual Machines	1	Yes [HP2005g]
IBM Logical Partitions	1	Yes [IBM2004a]
Microsoft Virtual Server 2005	1	Yes [MS2004c]
Solaris 10 Containers	1	Yes
		Comment: This can be achieved using a disk partition
VMware ESX Server 2.5	1	Yes [VMW2005a]
Fujitsu Extended Partitioning	1	Yes
		Comment: Dedicated disks
Fujitsu Physical Partitioning	1	Yes
		Comment: Dedicated disks
HP Node Partitions	1	Yes
		Comment: Dedicated disks
HP Virtual Partitions	1	Yes
		Comment: Dedicated disks
Sun Dynamic System Domains	1	Yes
		Comment: Dedicated disks

Table 6.12: Network Bandwidth Isolation (C 2.2.5)

Virtualisation System	*Rank*	*Network Bandwidth Isolation*
Fujitsu Extended Partitioning	1	Dedicated [FS2005a]
Fujitsu Physical Partitioning	1	Dedicated [FS2005a]
HP Node Partitions	1	Dedicated
HP Virtual Partitions	1	Dedicated
Solaris 10 Containers	1	Dedicated, scheduled
		Comment: (using IP QoS)
Sun Dynamic System Domains	1	Dedicated
VMware ESX Server 2.5	1	Dedicated, scheduled, scheduled capping
		Comment: Scheduling for outbound traffic only. [VMW2005a]. Alternative is dedicated adapter [IBM2005c]
HP Integrity Virtual Machines	2	Dedicated [HP2005g], unscheduled shared
IBM Logical Partitions	2	Dedicated, unscheduled shared
Microsoft Virtual Server 2005	2	Dedicated, unscheduled shared

Category Ranking

The resource isolation category rankings are shown in Figure 6.3. The physical partitioning systems provided the best isolation of resources, along with HP Virtual Partitions. HP Integrity Virtual Machines and Microsoft Virtual Server fared the worst. This was due to a lack of dedicated CPU support and scheduled network bandwidth guarantees. Refer to Table 6.8 and Table 6.12 for more details.

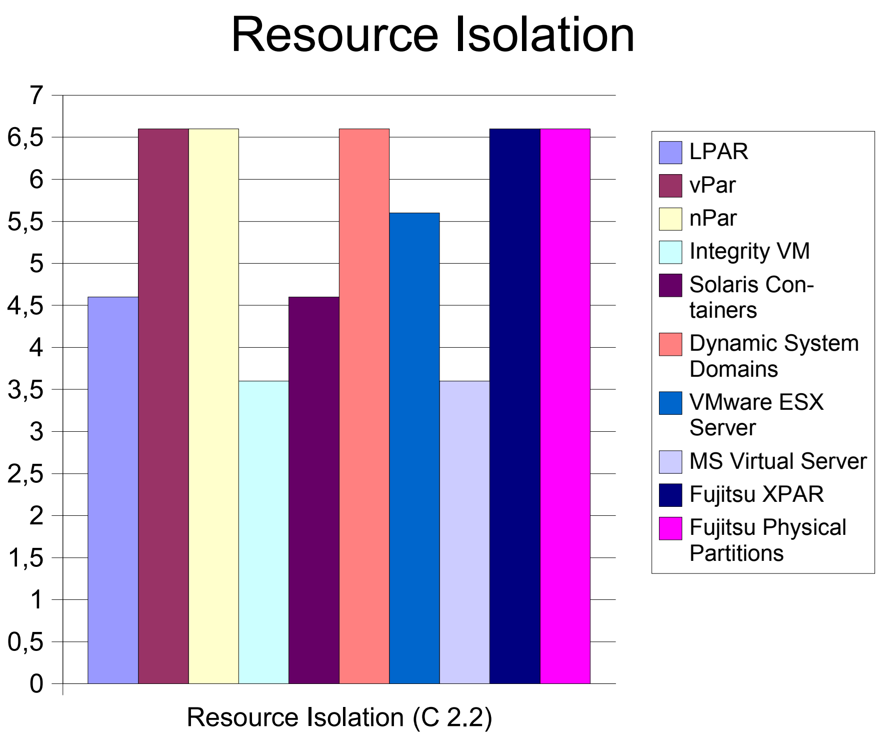


Figure 6.3: Ranking Results for the Resource Isolation Category

Security and Namespace Isolation (C 2.3)

Most of the criteria considered for the security and namespace isolation category are basic requirements for any virtualisation system. It is not surprising then that all of the systems evaluated satisfy most or all of the criteria. The evaluation results are presented in Tables 6.13 to 6.18.

Isolation of filesystems is supported by all systems. Solaris Containers only satisfies this requirement if no loopback mounts are used to share directories between partitions. This resulted in all the systems being ranked equally in Table 6.13.

All of the systems evaluated isolate information about installed packages and applications. For most systems this is due to the fact that each partition has a separate operating system instance. This result is recorded in Table 6.14.

Network port bindings are isolated by all of the systems, leaving applications in each partition free to bind to ports without being constrained by port bindings of applications in other partitions. This resulted in all of the systems being ranked equally in Table 6.15.

Inter-partition communication is restricted to standard network interfaces for all of the systems evaluated. Processes in one partition are unable to use any other communication channels. This information is recorded in Table 6.16.

Isolation of operating system tuning parameters, such as kernel parameters, is provided by the majority of the systems. Only Solaris Containers did not meet this criterion. The other systems, which create a separate operating system instance for each partition isolate these settings without any additional work. Solaris Containers provides support for isolating of some, but not all of these settings using projects. This is reflected in Table 6.17. This resulted in Solaris Containers being ranked below the other systems for this criterion.

All of the systems evaluated provide a separate set of users for each partition. This resulted in all of the systems being ranked equally in Table 6.18.

Table 6.13: Filesystem Isolation (C 2.3.1)

Virtualisation System	*Rank*	*Filesystem Isolation*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
		Comment: Only when no directories are shared via loopback mounted filesystem.
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Table 6.14: Isolation of Package Databases (C 2.3.2)

Virtualisation System	*Rank*	*Isolation of Package Databases*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Table 6.15: Isolation of Network Port Bindings (C 2.3.3)

Virtualisation System	*Rank*	*Isolation of Network Port Bindings*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Table 6.16: Inter-partition Communication Isolation (C 2.3.4)

Virtualisation System	*Rank*	*Inter-partition Communication Isolation*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Table 6.17: Isolation of Operating System Tuning Parameters (C 2.3.5)

Virtualisation System	*Rank*	*Isolation of Operating System Tuning Parameters*
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes
HP Integrity Virtual Machines	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
IBM Logical Partitions	1	Yes
Microsoft Virtual Server 2005	1	Yes
Sun Dynamic System Domains	1	Yes
VMware ESX Server 2.5	1	Yes
Solaris 10 Containers	2	No
		Comment: Some parameters are supported on a per-project basis, other are global and therefore not isolated.

Table 6.18: Separate Set of Users for Each Partition (C 2.3.6)

Virtualisation System	*Rank*	*Separate Set of Users for Each Partition*
IBM Logical Partitions	1	Yes
VMware ESX Server 2.5	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes
HP Virtual Partitions	1	Yes
HP Integrity Virtual Machines	1	Yes
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Category Ranking

The category ranking results depicted in Figure 6.4 reflect the fact that most of the systems scored maximum points for each of the criteria. The only exception is Solaris Containers which does not isolate all of the operating system tuning parameters. This system was ranked last in this category.

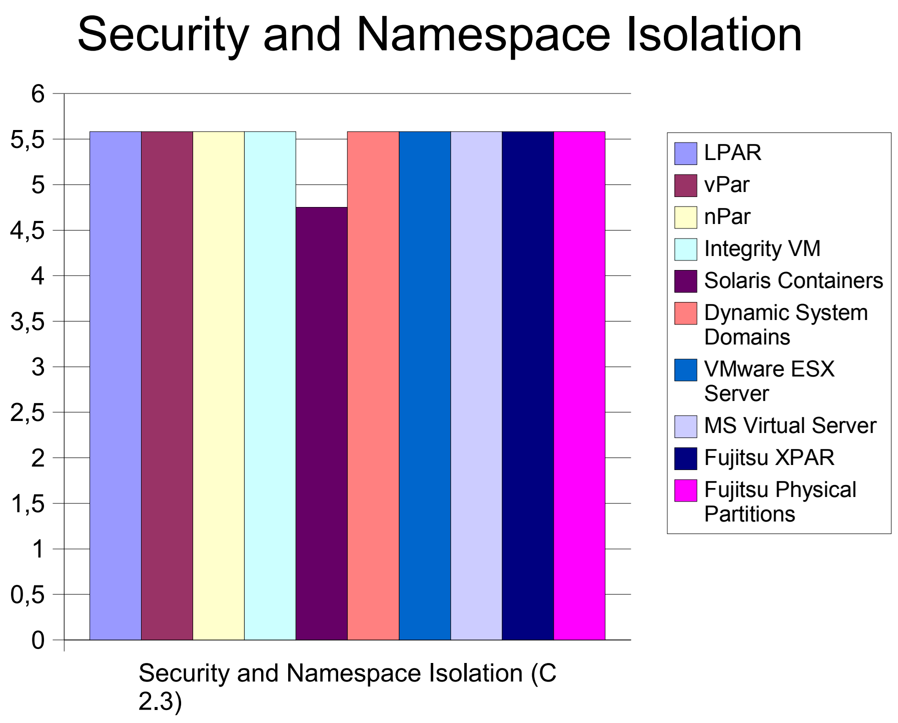


Figure 6.4: Ranking Results for the Security and Namespace Isolation Category

Manageability (C 3)

The manageability category of the framework concentrates on a small set of core manageability features. A result of this approach is that most of the systems evaluated in this section scored top points in most of the categories. Once these systems are also evaluated using the time taken to install partitions criterion, this will change. The manageability evaluation data is recorded in Tables 6.19 to 6.21.

Graphical management tools are available for all of the systems under consideration. This is reflected in Tables 6.19a and b. These systems are ranked equally for this criterion as a result. For some systems this is the default method for managing partitions. Other systems are typically managed using a command line interface (CLI), but also provide GUI tools. GUI management tools for Solaris Containers were only released a few months after the release of Solaris 10. The use of the new GUI tools is optional with many users, including the author, using only the command line utilities.

The management of partitions can be automated, using scripts, for all of the systems evaluated in this chapter. This is done using either standard Unix shell scripts or via special interfaces. Microsoft Virtual Server 2005 exposes a COM interface for managing partitions. This information is presented in Table 6.20.

As discussed in the introduction, it is not feasible to purchase and test an example of each system being evaluated here. Measuring the time it takes to install a new partition is an example of a criterion which cannot be evaluated by studying literature.

The only system supporting live migration of partitions between servers which was evaluated was VMware ESX Server. This resulted in this system being ranked ahead of all the other systems in Table 6.21.

Table 6.19a: GUI Tools for Partition Management (C 3.1)

Virtualisation System	***Rank***	***GUI Tools for Partition Management***
IBM Logical Partitions	1	Yes
		Comment: Hardware Management Console (HMC) panel
VMware ESX Server 2.5	1	Yes
		Comment: VMware Management Interface
Microsoft Virtual Server 2005	1	Yes
		Comment: Virtual Server Administration Web site [MS2004c]
Solaris 10 Containers	1	Yes
		Comment: (Optional - Solaris Container Manager 1.1 [SM2005a])
Sun Dynamic System Domains	1	Yes
		Comment: Solaris Management Center [SM2003a]

(continued...)

Table 6.19b: GUI Tools for Partition Management (C 3.1)

Virtualisation System	***Rank***	***GUI Tools for Partition Management***
HP Node Partitions	1	Yes
		Comment: HP Partition Manager [HP2005k]
HP Virtual Partitions	1	Yes
		Comment: GUI only available for PA-RISC servers, and only supported for HP-UX 11i v1 [HP2005h]. No GUI for Integrity servers, or PA-RISC servers running HP-UX 11i v2.
HP Integrity Virtual Machines	1	Yes [HP2005g]
Fujitsu Extended Partitioning	1	Yes
		Comment: Fujitsu ServerView [FS2004b]
Fujitsu Physical Partitioning	1	Yes
		Comment: Fujitsu ServerView [FS2004b]

Table 6.20: Scripting Support (C 3.2)

Virtualisation System	*Rank*	*Scripting Support*
IBM Logical Partitions	1	Yes [IBM2005b]
VMware ESX Server 2.5	1	Yes [VMW2005a]
Microsoft Virtual Server 2005	1	Yes [MS2004c]
Solaris 10 Containers	1	Yes [SM2005d]
Sun Dynamic System Domains	1	Yes
HP Node Partitions	1	Yes [HP2004c]
HP Virtual Partitions	1	Yes [HP2003a]
HP Integrity Virtual Machines	1	Yes [HP2005g]
Fujitsu Extended Partitioning	1	Yes
Fujitsu Physical Partitioning	1	Yes

Time Taken to Install a Partition (C 3.3)

This criterion requires each system to be tested individually. Results will vary, based on hardware configuration and results will have to be submitted on a case by case basis.

Table 6.21: Live Migration of Partitions Between Separate Servers (C 3.4)

Virtualisation System	*Rank*	*Live Migration of Partitions Between Separate Servers*
VMware ESX Server 2.5	1	Yes
		Comment: VMotion [VMW2005d]
Fujitsu Extended Partitioning	2	No
Fujitsu Physical Partitioning	2	No
HP Integrity Virtual Machines	2	No
HP Node Partitions	2	No
HP Virtual Partitions	2	No
IBM Logical Partitions	2	No
Microsoft Virtual Server 2005	2	No
Solaris 10 Containers	2	No
Sun Dynamic System Domains	2	No

Category Ranking

Support for live migration is the reason for ESX Server receiving the best overall ranking for this category. This can clearly be seen in Figure 6.5. Once the time taken to install partitions has been evaluated, very few systems will be ranked equally. The only reason for the equal rankings in the category is the limitations of a literature-based evaluation.

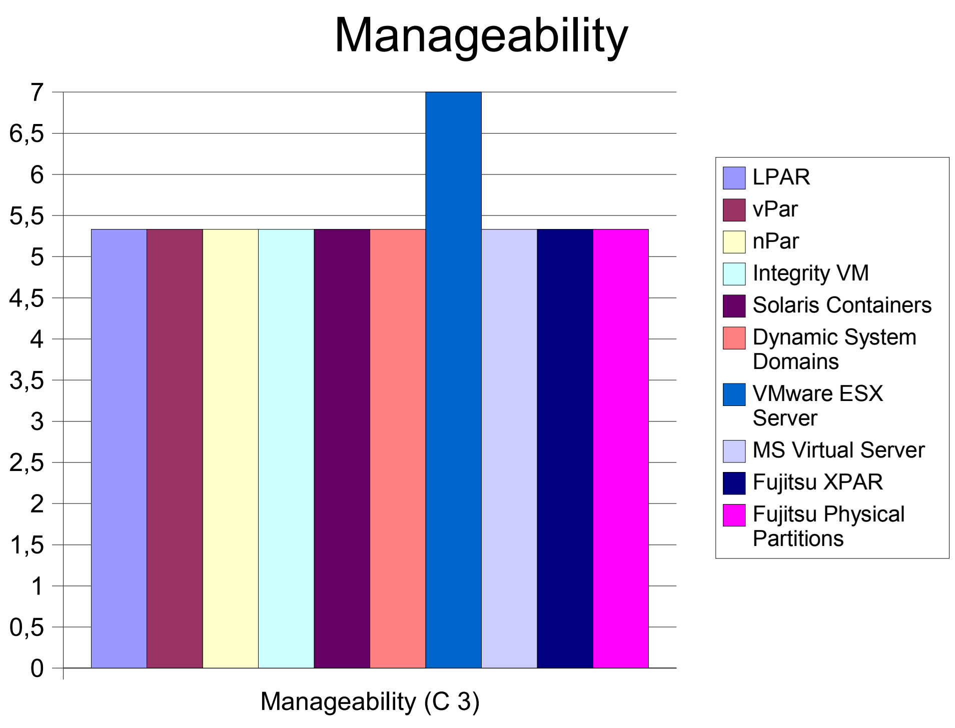


Figure 6.5: Ranking Results for the Manageability Category

Flexibility (C 4)

The focus of the flexibility category is on reallocation of system resources in order to respond to changing resource requirements. Systems which support dynamic resource reallocation fared best in this category.

Six of the ten systems evaluated support dynamic reallocation of CPUs, resulting in these systems being ranked in first place for this criterion. These included IBM Logical Partitions, Sun Dynamic System Domains, Solaris Containers and the Fujitsu partitioning systems, along with HP Virtual Partitions. It is worth noting that Virtual Partitions only supports reallocation of CPUs which cannot be used for I/O (unbound CPUs).

HP Node Partitions require affected partitions to be rebooted for CPUs to be reallocated. The same applies to the reallocation of virtual CPUs with HP Integrity Virtual Machines. Microsoft Virtual Server 2005 does not support CPU reallocation because it doesn't support more than a single CPU per partition. VMware ESX server requires a reboot to upgrade from one CPU to two. Downgrading back to a single CPU is not permitted. This information is summarised in Table 6.22.

Dynamic reallocation of memory was only supported by four of the ten systems evaluated. These systems were ranked in first place for this criterion. This feature is supported by Sun Dynamic System Domains, IBM Logical Partitions, Fujitsu Extended Partitions and Fujitsu Physical Partitions. This feature is supported for AIX, but not for Linux with LPARs. This is due to a lack of support for this type of reallocation by Linux currently. VMware ESX Server can vary memory allocations to a degree without rebooting using a balloon driver. This was excluded for reasons outlined in Chapter 5 where this criterion (C 4.2) was explained.

HP Node Partitions, HP Virtual Partitions and HP Integrity Virtual Machines all require partitions to be rebooted to reallocate memory. These systems were ranked in last place for this criterion as a result. The evaluations using this criterion are presented in Table 6.23.

There are tools to manage resource allocation based on system goals for HP Node Partitions, HP Virtual Partitions, HP Integrity Virtual Machines, IBM Logical Partitions and Solaris Containers. Of the systems studied, only nPartitions, vPartitions and Integrity Virtual Machines support resource management based on application goals. These systems were ranked in first place because they support both system and application goals. This is reflected in Tables 6.24a and b. Although workload management tools are one of the strengths of these systems, this needs to be considered in light of the limited facilities these systems provide for reallocation resources dynamically. As noted previously, none of the HP systems provide support for dynamic reallocation of memory. Dynamic CPU reallocation is supported for vPartitions (unbound CPUs only) but not nPartitions or Integrity Virtual Machines. Node Partitions provide the appearance of “moving” CPUs between partitions using iCoD (instant capacity upgrade on demand) CPUs. Processors in one partition are effectively deactivated in one partition while *different* idle CPUs are activated in another. The deactivated CPUs are then idle. CPU “migration” using capacity upgrade capabilities are excluded from the CPU reallocation without reboot criterion of the framework. Only systems which can migrate CPUs from one partition to another are considered to satisfy this criterion. The idea of capacity upgrade CPUs which are intentionally idle in a server, runs counter to one of the primary advantages of server virtualisation technology, namely that of improved resource utilisation.

Table 6.22: CPU Reallocation Without Reboot (C 4.1)

Virtualisation System	*Rank*	*CPU Reallocation Without Reboot*
Fujitsu Extended Partitioning	1	Yes [FS2004a]
Fujitsu Physical Partitioning	1	Yes [FS2004a]
HP Virtual Partitions	1	Yes
		Comment: Only unbound CPUs can be migrated without a reboot [HP2004k]. Each partition requires at least one bound CPU. Only bound CPUs can process I/O interrupts.
IBM Logical Partitions	1	Yes
Solaris 10 Containers	1	Yes
		Comment: Dynamic resource pools with processor sets [SM2005d].
Sun Dynamic System Domains	1	Yes
HP Integrity Virtual Machines	2	No
HP Node Partitions	2	No [HP2004b]
VMware ESX Server 2.5	2	No
		Comment: Cannot downgrade a VM from two virtual CPUs to one, upgrading a VM from one CPU to two requires the VM to be shut down [VMW2005a]. Multi-CPU support via virtual SMP add on.
Microsoft Virtual Server 2005	N/A	N/A
		Comment: A maximum of one CPU per partition.

Table 6.23: Memory Reallocation Without Reboot (C 4.2)

Virtualisation System	*Rank*	*Memory Reallocation Without Reboot*
Fujitsu Extended Partitioning	1	Yes [FS2004a]
Fujitsu Physical Partitioning	1	Yes [FS2004a]
IBM Logical Partitions	1	Yes
		Comment: This is not supported for Linux [BE2004]
Sun Dynamic System Domains	1	Yes
HP Integrity Virtual Machines	2	No
HP Node Partitions	2	No [HP2004c]
HP Virtual Partitions	2	No [HP2003a]
Microsoft Virtual Server 2005	2	No
		Comment: Similar to ESX server memory allocations are fixed – only variation via a balloon driver or change in maximum memory value which requires a reboot.
VMware ESX Server 2.5	2	No
		Comment: Memory allocations as seen by the operating system are fixed. This value corresponds to the maximum value assigned to a virtual machine. In practice a balloon driver is used to vary the actual quantity of physical memory available to each VM within this limit. Any changes to this maximum value requires a reboot.
Solaris 10 Containers	N/A	N/A
		Comment: All memory is pooled.

Table 6.24a: Tools to Automate Resource Allocation (C 4.3)

Virtualisation System	*Rank*	*Tools to Automate Resource Allocation*
HP Integrity Virtual Machines	1	System goal, application goal
		Comment: Global workload manager gWLM [HP2005l].
HP Node Partitions	1	System goal, application goal
		Comment: CPUs can be "moved" using HP-UX Workload Manager. In reality the CPUs are not moved as this is not supported by Node Partitions. CPUs in one partition can be deactivated and a corresponding number of spare CPUs in another can be activated. Only supports partitions running HP-UX [HP2004d].

(continued...)

Table 6.24b: Tools to Automate Resource Allocation (C 4.3)

Virtualisation System	*Rank*	*Tools to Automate Resource Allocation*
HP Virtual Partitions	1	System goal, application goal
		Comment: CPUs can be reallocated automatically using HP-UX Workload Manager [HP2004d].
IBM Logical Partitions	2	System goal
		Comment: Partition Load Manager only available for AIX. CPU and memory reallocation are supported [IBM2004a].
Solaris 10 Containers	2	System goal
		Comment: Using the dynamic resource pools with processor sets [SM2005d].
Fujitsu Extended Partitioning	3	None
Fujitsu Physical Partitioning	3	None
Microsoft Virtual Server 2005	3	None
Sun Dynamic System Domains	3	None
VMware ESX Server 2.5	3	None

Category Ranking

The ranking results for the flexibility category are depicted in Figure 6.6. HP Node Partitions, HP Virtual Partitions, HP Integrity Virtual Machines, VMware ESX Server and Microsoft Virtual Server 2005 provide poor support for dynamic resource reallocation. The rankings of the three HP systems improved considerably once support for workload management tools was taken into consideration. These systems were the only ones to support automated reallocation based on system *and* application goals. IBM Logical Partitions fared the best because it supports dynamic reallocation of both memory and CPUs, and provides workload management tools. Sun Dynamic System Domains and the Fujitsu partitioning systems provide support for dynamic resource reallocation of both CPUs and memory, but lack workload management tools. For this reason HP Virtual Partitions and Solaris Containers were able to achieve better rankings. The importance of dynamic memory reallocation versus workload management tools could be subject to some debate though.

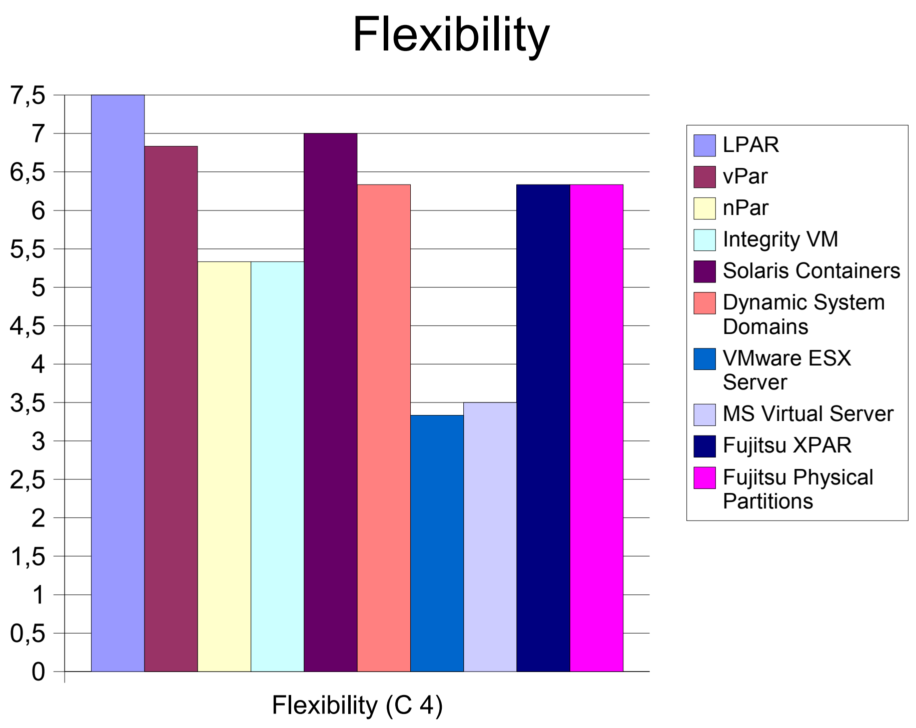


Figure 6.6: Ranking Results for the Flexibility Category

Granularity (C 5)

Granularity is a clear weakness for all of the physical partitioning systems evaluated. Software-based systems were ranked first for each of the individual criteria. These results are summarised in Tables 6.25 to 6.28.

Memory resource reallocation granularity varied widely between systems. Physical partitioning systems provide very coarse granularity, due to resources being allocated along physical boundaries. Granularity for the remaining systems was finer, but varied. The memory granularity of some systems depends on the quantity of memory in the server being partitioned. The results for this criterion are presented in Tables 6.25a and b.

Similarly, CPU reallocation granularity was also a weakness for physical partitioning systems. The level of memory granularity provided by these systems is very coarse. Systems which support shares or fractional allocations offer much finer granularity. VMware ESX Server, Microsoft Virtual Server, Solaris Containers, HP Integrity Virtual Machines and IBM Logical Partitions (when using shared processor partitions) provide a much finer degree of granularity. These systems were ranked highly as a result. Single CPU granularity is also provided by HP Virtual Partitions, Solaris Containers (using processor sets) and LPARs (using dedicated processor partitions). These results are summarised in Tables 6.26a and b.

Time granularity of memory and CPU reallocations depends on the specific equipment being used. This cannot be done without having actual systems installed on servers to test. Systems which do not support reallocation without rebooting (dynamic reallocation) are marked as not applicable for these criteria.

Table 6.25a: Memory Resource (Re)Allocation Quantity (C 5.1)

Virtualisation System	*Rank*	*Memory (Re)Allocation Quantity*
HP Integrity Virtual Machines	1	1 MB
		Comment: Memory allocations can be expressed in MB [HP2005g].
Microsoft Virtual Server 2005	1	1 MB
		Comment: Memory allocations usually expressed in MB.
VMware ESX Server 2.5	2	4 MB [VMW2005a]
IBM Logical Partitions	3	16 MB – 256 MB
		Comment: Granularity depends on the quantity of configurable physical memory in the system [IBM2005b] 16 MB if less than 4 GB 32 MB for up to 8 GB 64 MB for up to 16 GB 128 MB for up to 32 GB 256 MB for more than 32 GB

(continued...)

Table 6.25b: Memory Resource (Re)Allocation Quantity (C 5.1)

Virtualisation System	*Rank*	*Memory (Re)Allocation Quantity*
HP Virtual Partitions	4	64 MB [HP2004k]
Fujitsu Extended Partitioning	5	32 GB
		Comment: 32 GB for PrimePower 2500 Inferred from [FS2003a] and [FS2005b]. These numbers are for 2 GB DIMMs.
Sun Dynamic System Domains	5	32 GB
		Comment: All memory on a system board (up to 32 GB for all current systems) [SM2003b]
Fujitsu Physical Partitioning	6	64 GB
		Comment: All memory on a system board. This figure is for PrimePower 900/1500/2500. Inferred from [FS2003a] and [FS2005b]. This number is for 2 GB DIMMs.
HP Node Partitions	6	64 GB
		Comment: All memory on a cell board. 64 GB per board using 2 GB DIMMs [HP2005j].
Solaris 10 Containers	N/A	N/A
		Comment: All memory is pooled.

Table 6.26a: CPU Resource (Re)Allocation Quantity (C 5.2)

Virtualisation System	*Rank*	*CPU Resource (Re)Allocation Quantity*
Microsoft Virtual Server 2005	1	0.001 CPU
		Comment: Very fine granularity (shares based) [MS2004c], so a value of 0.001 is recorded.
Solaris 10 Containers	1	0.001 - 1 CPU [SM2005d]
		Comment: Very fine granularity (shares based) with the fair share scheduler [SM2005d], so a value of 0.001 is recorded.
VMware ESX Server 2.5	1	0.001 CPU
		Comment: Very fine granularity (shares based) [VMW2005a], so a value of 0.001 is recorded.
HP Integrity Virtual Machines	2	0.01 CPU
		Comment: 1/100th of a CPU

(continued...)

Table 6.26b: CPU Resource (Re)Allocation Quantity (C 5.2)

Virtualisation System	*Rank*	*CPU Resource (Re)Allocation Quantity*
IBM Logical Partitions	2	0.01 – 1 CPU
		Comment: 1/100th of a CPU for shared processor partitions [IBM2005b]. 1 CPU for dedicated processor partitions.
HP Virtual Partitions	3	1 CPU [HP2004b]
Fujitsu Extended Partitioning	4	2 – 4 CPUs
		Comment: 2 CPUs for PrimePower 900/1500 4 CPUs for PrimePower 2500 [FS2003a]
HP Node Partitions	5	4 – 8 CPUs
		Comment: All CPUs on a system board (up to 4 or 8, depending on server model) [HP2004b]. 8 CPUs per system board for dual CPU modules.
Sun Dynamic System Domains	5	4 – 8 CPUs
		Comment: All CPUs on a system board (up to 4) [SM2001]. 8 cores for UltraSPARC IV and newer.
Fujitsu Physical Partitioning	6	8 CPUs
		Comment: 8 CPUs for PrimePower 900/1500/2500 [FS2003a]

Table 6.27: CPU Reallocation Time Granularity (C 5.3)

Virtualisation System	*CPU Reallocation Time Granularity*
IBM Logical Partitions	Requires experimentation to determine
VMware ESX Server 2.5	N/A
	Comment: requires reboot
Microsoft Virtual Server 2005 (shares)	N/A
	Comment: Maximum of one CPU per partition.
Solaris 10 Containers	Requires experimentation to determine
Sun Dynamic System Domains	Requires experimentation to determine
HP Node Partitions	N/A (requires reboot)
HP Virtual Partitions	Requires experimentation to determine
HP Integrity Virtual Machines	N/A (requires reboot)
Fujitsu Extended Partitioning	Requires experimentation to determine
Fujitsu Physical Partitioning	Requires experimentation to determine

Table 6.28: Memory Reallocation Time Granularity (C 5.4)

Virtualisation System	*Memory Reallocation Time Granularity*
IBM Logical Partitions	Requires experimentation to determine
VMware ESX Server 2.5	N/A
	Comment: Not dynamic – change in maximum memory value requires reboot.
Microsoft Virtual Server 2005	N/A
	Comment: Requires reboot
Solaris 10 Containers	N/A
	Comment: All memory is pooled.
Sun Dynamic System Domains	Requires experimentation to determine
HP Node Partitions	N/A
	Comment: not dynamic (requires reboot)
HP Virtual Partitions	N/A
	Comment: not dynamic (requires reboot)
HP Integrity Virtual Machines	Requires experimentation to determine
Fujitsu Extended Partitioning	Requires experimentation to determine
Fujitsu Physical Partitioning	Requires experimentation to determine

Category Ranking

The ranking results for this category are depicted in Figure 6.7. The poor granularity of physical partitioning systems can clearly be seen in this chart. Microsoft Virtual Server 2005 and Solaris Containers provide the finest level of granularity.

Although the author does not have the resources to measure the time granularity for all of the systems, it is unlikely that this information will significantly affect the rankings. This is because physical partitioning systems are also likely to provide the coarsest time granularity.

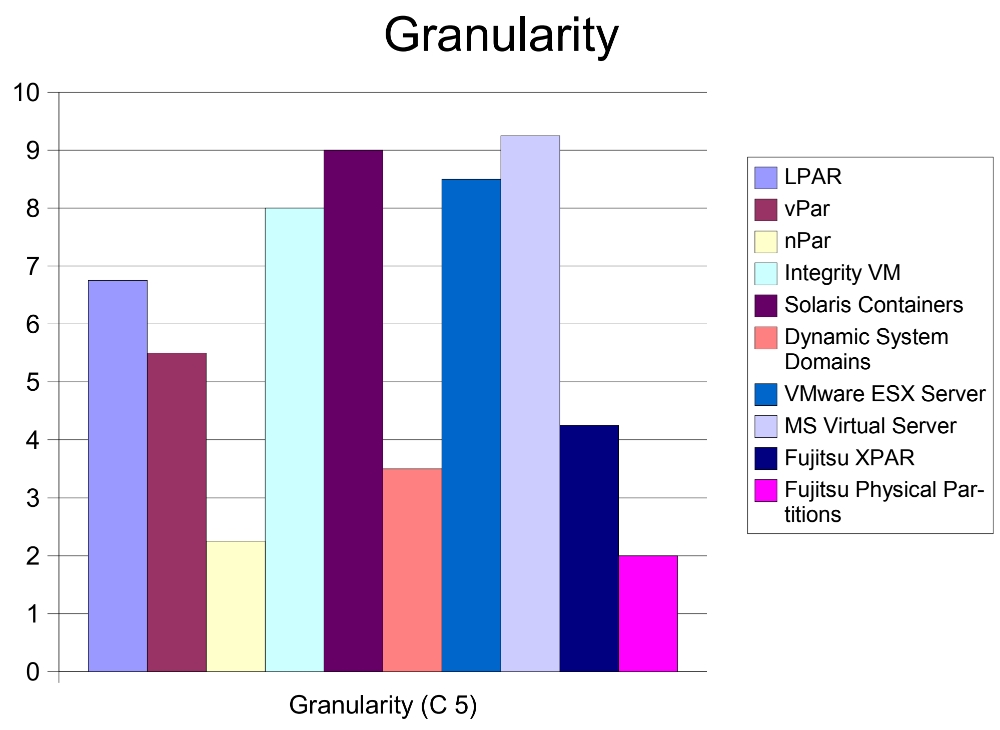


Figure 6.7: Ranking Results for the Granularity Category

Scalability – Scaling to Larger Partitions (C 6.1)

Systems which were designed for large servers fared the best in this category. Smaller systems such as Microsoft Virtual Server 2005, VMware ESX Server do not scale to a large number of CPUs per partition. These systems restrict each partition to a maximum of one or two CPUs respectively. HP Integrity Virtual Machines limit each partition to a maximum of four CPUs.

All of the physical partitioning systems as well as IBM Logical Partitions, HP Virtual Partitions and Solaris Containers provide support for a large number of CPUs and large quantities of memory. These systems were ranked highly in Tables 6.29 and 6.30 as a result.

HP Node Partitions have a per partition CPU limit which is operating system dependent. Only HP-UX can be used in a 128 CPU partition. Refer to Tables 6.29a, b and c for more information on the number of CPUs supported by each system.

Similarly, Microsoft Virtual Server 2005 and VMware ESX Server do not provide support for large quantities of memory in a partition. Other systems provide support for much larger memory allocations. The maximum per-partition memory allocations are recorded in Tables 6.30 and b.

Table 6.29a: Maximum CPUs Per Partition (C 6.1.1)

Note: The number of CPUs in the largest available server known to support the system was used for cases where no documented per-partition CPU limit for a system was found.

Virtualisation System	*Rank*	*Maximum CPUs Per Partition*
Solaris 10 Containers	1	144
		Comment: Sun Fire E25k supports up to 144 processor cores [SM2005c] (72 x dual core).
Sun Dynamic System Domains	1	144
		Comment: Sun Fire E25k supports up to 144 processor cores [SM2005c] (72 x dual core).
Fujitsu Physical Partitioning	2	128 [FS2005a]
		Comment: All of the CPUs in a fully configured PrimePower 2500 system.

(continued...)

Table 6.29b: Maximum CPUs Per Partition (C 6.1.1)

Virtualisation System	*Rank*	*Maximum CPUs Per Partition*
HP Node Partitions	2	8 - 128
		Comment: 128 CPUs supported on largest superdome system [HP2004c]. This requires HP-UX B11.23 September 2004 version or later [HP2005i]. 64 CPUs supported for Windows 2003 Server [HP2005i]. 8 processors (maximum of two cells, and HP mx2 dual-processor modules are not supported) supported for Red Hat Enterprise Linux 3 Updates 2 and 3 [HP2005i]. 16 processors supported for SuSE Linux Enterprise Server 9 and OpenVMS (dual-processor modules not supported) [HP2005i].

(continued...)

Table 6.29c: Maximum CPUs Per Partition (C 6.1.1)

Virtualisation System	*Rank*	*Maximum CPUs Per Partition*
HP Virtual Partitions	2	128
		Comment: Assuming all CPUs of a superdome can be added to a virtual partition
Fujitsu Extended Partitioning	3	64 [FS2005a]
		Comment: Only half of the PrimePower 2500 system boards support Extended Partitioning.
IBM Logical Partitions	3	64
		Comment: Largest pSeries server p5-595 [IBM2005f] currently has 64 CPU cores.
HP Integrity Virtual Machines	4	4 [HP2005g]
VMware ESX Server 2.5	5	2 [VMW2005a]
		Comment: With Virtual SMP, otherwise the limit is one CPU. Virtual SMP only supported on Linux and Windows.
Microsoft Virtual Server 2005	6	1 [MS2004c]

Table 6.30a: Maximum Memory Per Partition (C 6.1.2)

Virtualisation System	*Rank*	*Maximum Memory Per Partition*
IBM Logical Partitions	1	2 TB [IBM2005f]
		Comment: IBM p5 595 supports up to 2 TB of memory.
HP Integrity Virtual Machines	2	1 TB
		Comment: No memory limit in documentation, so memory of largest server recorded.
HP Node Partitions	2	96 GB – 1 TB
		Comment: 1 TB [HP2005e] using HP-UX B11.23 September 2004 version or later or Windows 2003 Server. 96 GB limit for Red Hat Enterprise Linux 3 Update 2 [HP2005i] 128 GB limit for Red Hat Enterprise Linux 3 Update 3 [HP2005i] 256 GB limit for SuSE Linux Enterprise Server 9 [HP2005i]

(continued...)

Table 6.30b: Maximum Memory Per Partition (C 6.1.2)

Virtualisation System	*Rank*	*Maximum Memory Per Partition*
HP Virtual Partitions	2	1 TB [HP2005e]
		Comment: No memory limit in documentation, so memory of largest server recorded.
Solaris 10 Containers	3	576 GB
		Comment: Maximum memory for a Sun Fire E25K [SM2005c].
Sun Dynamic System Domains	3	576 GB
		Comment: Maximum memory for a Sun Fire E25K [SM2005c].
Fujitsu Extended Partitioning	4	512 GB
		Comment: 512 GB supported by PrimePower 2500 [FS2005b].
Fujitsu Physical Partitioning	4	512 GB
		Comment: 512 GB supported by PrimePower 2500 [FS2005b].
Microsoft Virtual Server 2005	5	3.6 GB
VMware ESX Server 2.5	5	3.6 GB
		Comment: 3.6 GB per virtual machine limit [IBM2005c]. Maximum of 64 GB for whole system [VMW2004e].

Category Ranking

The ranking results for the scalability – scaling to larger partitions category are depicted in Figure 6.8. The scalability advantages of systems designed for larger servers are clear. The only anomaly is the high score of Integrity Virtual machines. This system currently limits each partition to a maximum of four CPUs, but supports large quantities of memory. This resulted in a score similar to that of Fujitsu Physical Partitions. The reason for this is that an Integrity VM can be allocated twice as much memory as a Fujitsu Physical Partition. Fujitsu Physical Partitions support up to 128 CPUs compared to the four of Integrity VMs. This is most likely a result of the relative simplicity of ranking model adopted.

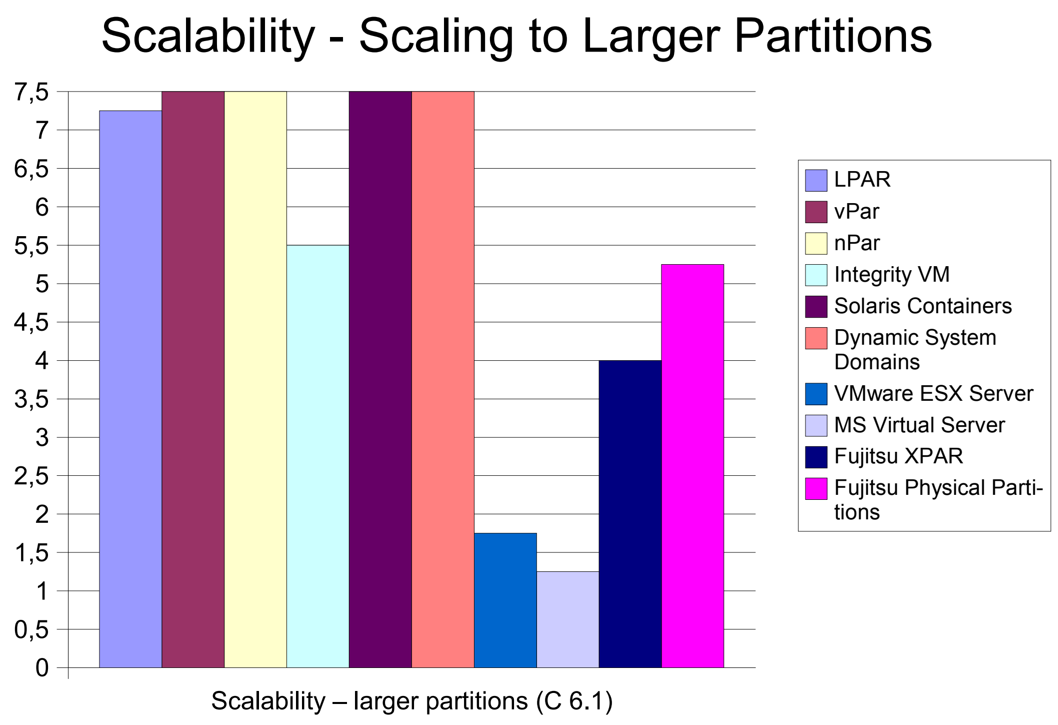


Figure 6.8: Ranking Results for the Scalability – Scaling to Larger Partitions Category

Scalability – Scaling to More Partitions (C 6.2)

The importance of scaling to a large number of partitions was discussed in Chapter 5. In most cases this depends on the quantity of resources consumed by each partition. This is reflected in the criteria for this category. The evaluations for these criteria are summarised in Tables 6.31 to 6.37.

For some systems the maximum number of partitions per CPU is restricted to a specific value. IBM Logical Partitions limits the number of partitions per CPU to ten. For VMware ESX Server the limit is eight. The physical partitioning systems evaluated have to allocate multiple CPUs to each partition. At the opposite end of the scale Microsoft Virtual Server does not impose any such limit. Solaris Containers supports a maximum of 8192 partitions per server, irrespective of the number of CPUs. Resource consumption is likely to impose a lower limit on all but the largest servers. Microsoft Virtual Server and Solaris Containers are ranked first and second place for this criterion as a result. This information is recorded in Table 6.31.

Sharing disks and network adapters between partitions removes limits imposed by the number of adapters available. Five of the ten systems evaluated support sharing of disks and network adapters. These systems are ranked in first place in Tables 6.32 and 6.33 as a result. The four physical partitioning systems evaluated and HP Virtual Partitions only support dedicating devices to partitions.

Most of the virtualisation systems evaluated in this section host multiple operating system instances, resulting in heavy memory consumption. System virtual machine monitors and hypervisors also result in additional memory overheads. These overheads are better documented for some systems than others. This contributes towards overall memory consumption of partitions. The actual memory usage depends on a number of factors and is best evaluated by installing and using each of the systems. Some information about memory usage and overhead can be gleaned from documentation. This is discussed in the comments section of Tables 6.34a, b and c.

Disk space consumed by each partition is also larger for systems which require an entire operating system installation. In order to measure the disk space usage an example of each system would need to be purchased and tested. This is indicated in Tables 6.35a and b. Solaris Containers shares a single operating system instance between partitions which should greatly reduce disk space usage. The actual disk space usage depends on how many folders, if any, are shared with the global zone via loopback mounts.

Solaris Containers is the only system evaluated which pools memory between partitions. If memory usage capping (discussed in Chapter 4) is not enforced, the resultant pooling effect eliminates the silo effect of fixed memory resource allocations. This resulted in Solaris Containers being ranked in first place for this criterion. Table 6.36 reflects this.

Most of the systems under consideration also have a fixed upper limit on the number of partitions per system. For IBM Logical Partitions this limit is 254 partitions. For larger systems this limit is more restrictive than the ten partitions per CPU limit mentioned earlier. Similarly, VMware ESX server supports a maximum of 80 partitions per server, despite the previously stated limit of eight partitions per CPU (ESX Server can be used on servers with up to 16 CPUs). For physical partitioning systems the limit is hardware dependent. Solaris Containers has an upper limit of 8192 partitions per system, whereas no limit is specified for Microsoft Virtual Server, HP Integrity Virtual Machines and HP Virtual Partitions. These systems were ranked in first place for this criterion. For more information refer to Tables 6.37a and b.

Table 6.31: Maximum Partitions Per CPU (C 6.2.1)

Virtualisation System	*Rank*	*Maximum Partitions Per CPU*
Microsoft Virtual Server 2005	1	No limit in documentation
Solaris 10 Containers	2	8192
		Comment: 8192 is a per system [SM2005d] limit, so 8192 would be the theoretical per CPU maximum for a system with a single CPU..
HP Integrity Virtual Machines	3	20
		Comment: Each partition must be allocated at least a 5% CPU guarantee [HP2005g].
IBM Logical Partitions	4	10 [IBM2004a]
VMware ESX Server 2.5	5	8 [VMW2004e]
HP Virtual Partitions	6	1
Fujitsu Extended Partitioning	7	0.25 – 0.5
HP Node Partitions	8	0.25 Minimum of one cell board per partition
Sun Dynamic System Domains	8	0.25 Minimum of one system board per partition
Fujitsu Physical Partitioning	9	0.125 Minimum of one system board per partition

Table 6.32: Sharing of Network Adapters (C 6.2.2)

Virtualisation System	*Rank*	*Sharing of Network Adapters*
HP Integrity Virtual Machines	1	Yes [HP2005g]
IBM Logical Partitions	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
VMware ESX Server 2.5	1	Yes
Fujitsu Extended Partitioning	2	No
Fujitsu Physical Partitioning	2	No
HP Node Partitions	2	No
HP Virtual Partitions	2	No
Sun Dynamic System Domains	2	No

Table 6.33: Sharing of Disk Resources (Disk I/O Adapters and Local Disks) (C 6.2.3)

Virtualisation System	*Rank*	*Sharing of Disk Resources (Disk I/O Adapters and Local Disks)*
HP Integrity Virtual Machines	1	Yes
IBM Logical Partitions	1	Yes
Microsoft Virtual Server 2005	1	Yes
Solaris 10 Containers	1	Yes
VMware ESX Server 2.5	1	Yes
Fujitsu Extended Partitioning	2	No
Fujitsu Physical Partitioning	2	No
HP Node Partitions	2	No
HP Virtual Partitions	2	No
Sun Dynamic System Domains	2	No

Table 6.34a: Memory Consumption: Memory Usage with Fifteen Partitions Booted (C 6.2.4)

Virtualisation System	***Memory Consumption: Memory Usage with Fifteen Partitions Booted***
IBM Logical Partitions	*Needs to be measured experimentally*
	Comment: Includes memory consumed by operating system instances and any overhead. The overhead component of the memory consumption can be calculated using IBM's LPAR Validation Tool [IBM2005h]. Memory overhead depends on a number of factors [IBM2005b].
VMware ESX Server 2.5	*Needs to be measured experimentally*
	Comment: Includes memory consumed by the operating system instances installed in the partitions, plus any other memory overhead due to virtualisation. Memory consumption can be reduced using page sharing [Wal2002]. Per-partition overhead will be at least 54 MB [VMW2005a]. The VMM and service console also consume memory.
Microsoft Virtual Server 2005	*Needs to be measured experimentally*
	Comment: Includes memory consumed by the operating system instances installed in the partitions, plus any other memory overhead due to virtualisation. A 32 MB per-VM overhead which is claimed by Microsoft [MS2004d]. Memory consumed by host OS is also included.

(continued...)

Table 6.34b: Memory Consumption: Memory Usage with Fifteen Partitions Booted (C 6.2.4)

Virtualisation System	*Memory Consumption: Memory Usage with Fifteen Partitions Booted*
Solaris 10 Containers	*Needs to be measured experimentally*
	Comment: Should be lower than other systems that host a separate operating system instance in each partition. Will include per-container memory consumption plus memory consumption of global Solaris instance.
Sun Dynamic System Domains	*Needs to be measured experimentally*
	Comment: Depends on the memory usage of the operating system installed in the partitions.
HP Node Partitions	*Needs to be measured experimentally*
	Comment: Depends on the memory usage of the operating system installed in the partitions.
HP Virtual Partitions	*Needs to be measured experimentally*
	Comment: Memory consumed by the operating system instance installed in the partitions, plus any other memory overhead due to virtualisation. Minimum memory required for HP-UX is between 256 MB and 512 MB depending on the server model [HP2004f].

(continued...)

Table 6.34c: Memory Consumption: Memory Usage with Fifteen Partitions Booted (C 6.2.4)

Virtualisation System	***Memory Consumption: Memory Usage with Fifteen Partitions Booted***
HP Integrity Virtual Machines	*Needs to be measured experimentally*
	Comment: For each guest 7% of memory configured will be overhead according to Hewlett Packard [HP2005g]. VMM is claimed to consume 750 MB of memory [HP2005g]. Will also include memory consumed by OS instances in partitions.
Fujitsu Extended Partitioning	*Needs to be measured experimentally*
	Comment: Depends on the memory usage of the operating system installed in the partition.
Fujitsu Physical Partitioning	*Needs to be measured experimentally*
	Comment: Depends on the memory usage of the operating system installed in the partition.

Table 6.35a: Disk Space Consumed by a Partition (C 6.2.5)

Virtualisation System	*Disk Space Consumed by a Partition*
IBM Logical Partitions	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
VMware ESX Server 2.5	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
Microsoft Virtual Server 2005	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
Solaris 10 Containers	*Needs to be measured experimentally*
	Comment: Depends on packages installed [SM2005d].
Sun Dynamic System Domains	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed

(continued...)

Table 6.35b: Disk Space Consumed by a Partition (C 6.2.5)

Virtualisation System	***Disk Space Consumed by a Partition***
HP Node Partitions	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
HP Virtual Partitions	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
HP Integrity Virtual Machines	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed. Host consumes 50 MB of disk space [HP2004d]
Fujitsu Extended Partitioning	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed
Fujitsu Physical Partitioning	*Needs to be measured experimentally*
	Comment: Depends on disk space consumed by operating system installed

Table 6.36: Pooling of Memory (C 6.2.6)

Virtualisation System	*Rank*	*Pooling of Memory*
Solaris 10 Containers	1	Yes
Fujitsu Extended Partitioning	2	No
Fujitsu Physical Partitioning	2	No
HP Integrity Virtual Machines	2	No
HP Node Partitions	2	No
HP Virtual Partitions	2	No
IBM Logical Partitions	2	No
Microsoft Virtual Server 2005	2	No
Sun Dynamic System Domains	2	No
VMware ESX Server 2.5	2	No

Table 6.37a: Fixed Limits on Number of Partitions Per System (C 6.2.7)

Virtualisation System	*Rank*	*Fixed Limits on Number of Partitions Per System*
HP Integrity Virtual Machines	1	No
		Comment: Limited only by hardware resources
HP Virtual Partitions	1	No
Microsoft Virtual Server 2005	1	No
		Comment: Host server can have up to 32 CPUs [MS2004c].
Solaris 10 Containers	2	8192
		Comment: Maximum of 8192 partitions per server [SM2005d].
IBM Logical Partitions	3	254
		Comment: Maximum of 254 partitions per server. This is limiting factor for 32-way systems and higher. This overrides the 10 partitions per CPU limit for micro partitioning.

(continued...)

Table 6.37b: Fixed Limits on Number of Partitions Per System (C 6.2.7)

Virtualisation System	*Rank*	*Fixed Limits on Number of Partitions Per System*
VMware ESX Server 2.5	4	80
		Comment: Maximum of 80 partitions per server [VMW2004e]. Limit is 40 if virtual SMP is used to allocate 2 virtual CPUs to each partition. Host server can have a maximum of 16 CPUs [IBM2005c]
Sun Dynamic System Domains	5	18
		Comment: Limited by number of I/O boards. This is dependent on the server. On a Sun Fire E15K and E25K [SM2005c] the limit is 18 domains [SM2003b].
HP Node Partitions	6	16 [HP2004c]
Fujitsu Extended Partitioning	7	8 - 15
		Comment: 8 for PrimePower 900 15 for PrimePower 1500/2500 [FS2005b]
Fujitsu Physical Partitioning	7	2 - 15
		Comment: 2 for PrimePower 900 4 for PrimePower 1500 15 for PrimePower 2500 [FS2005b]

Category Ranking

The ranking results for the scalability – scaling to more partitions category are depicted in Figure 6.9. Solaris Containers is ranked the highest as it is the most lightweight of the virtualisation systems evaluated. Microsoft Virtual Server and HP Integrity Virtual Machines also fared well. The physical partitioning systems were ranked the lowest. This is due to the requirement for resources to be physically dedicated to partitions. Although memory consumption and disk space usage could not be measured for all systems this is unlikely to significantly affect the results. If anything it will further benefit the ranking of Solaris Containers.

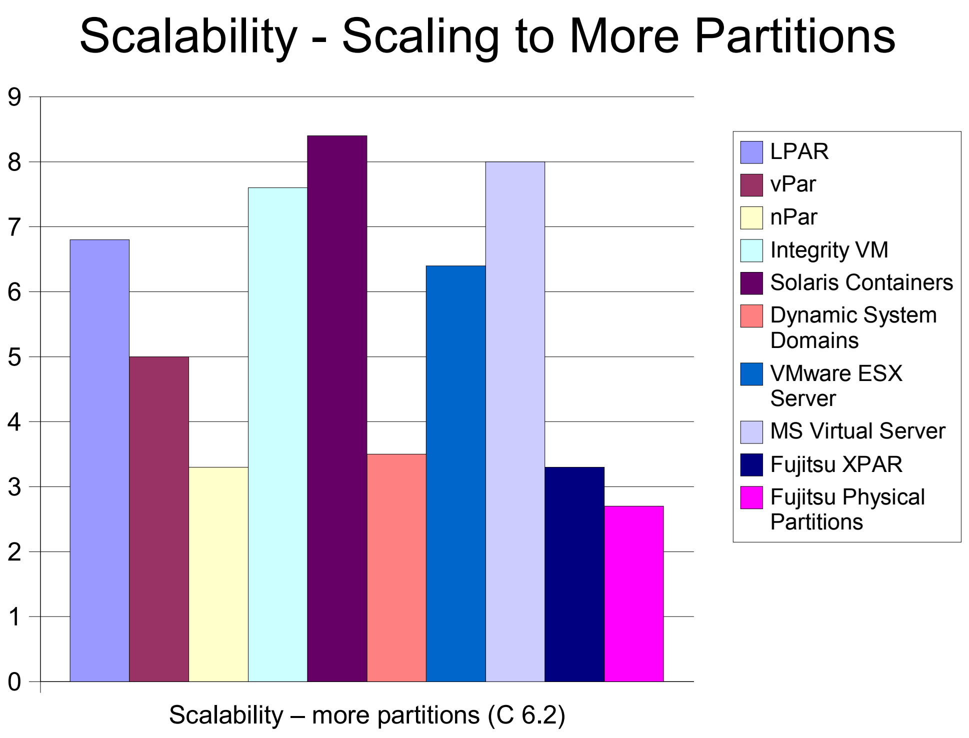


Figure 6.9: Ranking Results for the Scalability – Scaling to More Partitions Category

Performance (C 7)

Evaluating performance will require purchasing an example of each system, which is not feasible as was stated previously. Results are dependent on the actual system hardware and software configuration used. Refer to Chapter 7 for an example of an evaluation including the performance criteria.

6.3 Strengths and Weaknesses Analysis

One of the stated goals of the framework was to have enough criteria to be able to highlight both the strengths and weaknesses of each system. Based on the evaluation conducted in this chapter, what follows is a brief summary of some of the strengths and weaknesses identified for the systems evaluated. Bear in mind that the preceding evaluations did not include all of the criteria. This is especially true for the performance criteria. Identifying strengths and weaknesses was still feasible because the majority of the evaluation criteria were used in these evaluations.

IBM Logical Partitions

Strengths

Logical Partitions support scalability in both directions, faring well in both of the scalability categories. Operating system compatibility was also strong with LPARs supporting three different operating systems. This system also supports larger Linux partitions than any of the other systems evaluated. Resources can be reassigned at a reasonably fine degree of granularity. CPUs can be dedicated to partitioned or shared. Flexibility was a clear strength. Both memory and CPUs can be reallocated dynamically, although dynamic memory reallocation is not supported by Linux. Resources reallocation between partitions can be automated based on system goals. Sharing of network adapters and disk resources between partitions is supported.

Weaknesses

Only the POWER ISA is supported, limiting compatibility to this type of hardware. Tools to automate resource reallocation are only available for AIX and dynamic reallocation is not supported for Linux. This system does not offer physical isolation from hardware failures. Disk and network bandwidth are not supported for shared devices. Memory overhead due to the hypervisor can reduce the number of partitions possible on smaller servers. Memory granularity depends on the quantity of memory configured. For larger configurations this granularity is quite coarse.

VMware ESX Server 2.5

Strengths

ESX Server was the only system evaluated which offered live migration of partitions (virtual machines) between separate physical servers. It is also the only system which provides support for sharing of duplicate pages between partitions. Resources can be allocated at a fine granularity. Compatibility with a wide range of x86 operating systems was a strong point for this system. ESX server can also scale to a large number of partitions. This system was the only system evaluated which provides disk bandwidth guarantees for shared disks. It was also one of only two systems to support network bandwidth guarantees for shared network adapters.

Weaknesses

ESX server is limited to x86-based systems. Scalability is limited – a maximum of two CPUs and 3.6 GB of memory per partition. Dynamic memory and CPU reallocation are not supported (although physical memory usage can be managed to a limited degree using a balloon driver). Memory usage overhead is quite high due to the use of a virtual machine monitor. Physical isolation from hardware failures is not provided. Hardware support is limited by restricting support to a set of supported devices.

Microsoft Virtual Server 2005

Strengths

Resources can be allocated at a fine granularity. Microsoft Virtual Server can also scale to a large number of partitions, limited only by available resources.

Weaknesses

MS Virtual Server cannot scale to large partitions. Scalability limited to a maximum of one CPU and 3.6 GB of memory per partition. No dynamic memory or CPU reallocation, although lack of CPU reallocation is irrelevant as there is only support for one CPU per partition. This system does not provide resource isolation for either disk or network bandwidth. Physical isolation from hardware failures is not supported. Only one operating system, Windows, is officially supported currently. The only ISA supported is x86. Isolation between guest operating system instances depends on the stability of the host operating system as this is a dual mode hosted VMM system. This also results in a memory overhead for each partition. This design is likely to result in relatively poor performance, but this would need to be confirmed experimentally. For more information about system VMMs refer to Chapter 3.

Solaris 10 Containers

Strengths

Solaris Containers provides support for scaling in both directions. Due to low memory consumption and the container-based approach adopted, this system will scale to a large number of partitions per server. This was the only system supporting x86 which scales beyond two CPUs per partition. Resources can be allocated at a very fine level granularity and dynamic reallocation of CPUs is supported. A tool to dynamically reallocate CPUs in response to system demand is also provided. Solaris Containers supports both x86 and SPARC based systems, providing compatibility with servers of widely varying capability. Memory and disk space usage are lower than for systems which host multiple operating system instances. The performance overhead of this system is almost certainly lower than that of any of the other systems evaluated (excluding the physical partitioning systems), given the fact that this was the only container-based system being evaluated. Performance overhead is evaluated in greater detail in Chapter 7. This was one of only two systems to support network bandwidth guarantees for shared network adapters.

For larger systems Solaris Containers cannot be viewed in isolation. Containers can be used in conjunction with Sun Dynamic System Domains or any of the Fujitsu Partitioning systems evaluated which provide hardware fault isolation and support for multiple operating system instances. Each domain can host a different version of

Solaris. Solaris Containers can subdivide domains running Solaris 10. Thus Solaris Containers and Dynamic System Domains can be used together, with boundaries depending on the degree of isolation desired.

Weaknesses

Solaris 10 is the only operating system supported. A single Solaris instance is shared between all containers, so each container uses the same version of Solaris and the same patch level, limiting compatibility. Full isolation of operating system tuning parameters is not provided. Dedicating memory to partitions is also not supported. Only memory usage capping, which is enforced asynchronously, is provided. Fault isolation is low due to a single operating system instance being shared between all containers. In practice the level of fault isolation will depend on the stability of the Solaris 10 operating system.

Sun Dynamic System Domains

Strengths

Dynamic System Domains provide hardware fault isolation. Support for dynamic reallocation of CPUs and memory is another strength of this system. Physical isolation also provides strong isolation of resources. Domains enable multiple versions of Solaris to be hosted simultaneously on the same system. Scaling to very large partitions is also supported. As mentioned previously Dynamic System Domains can be used in conjunction with Solaris Containers.

Weaknesses

Resource allocation granularity is very coarse – at a system board level. Solaris is the only operating system supported and SPARC is the only ISA supported by this system. There are no tools to automate reallocation of resources in response to system demand. The number of domains per system is fairly limited due to the granularity of resource allocations. Dynamic System Domains are only supported on larger SPARC based systems.

HP Node Partitions

Strengths

Node Partitions provide hardware fault isolation with support for four different operating systems. These include HP-UX (Itanium and PA-RISC), Windows (Itanium), Red Hat Linux (Itanium) and OpenVMS. Both PA-RISC and Itanium hardware is supported. Node Partitions can scale many CPUs and large quantities of memory. HP Workload Manager can be used to automate resource reallocation in response to either application or system demands. This is of marginal value for this system, as the only resource which can be reallocated between Node Partitions is instant capacity on demand (iCoD) CPUs. In reality these CPUs are not actually moved between partitions. For more information refer to the discussion in the flexibility (C 4) section of this chapter. Virtual Partitions or Integrity Virtual Machines can be run within a Node Partition providing a finer granularity within Node Partitions.

Weaknesses

Node Partitions have operating system dependent sizing limits. Only Node Partitions running HP-UX support over 64 CPUs per partition. Dynamic reallocation of CPUs and memory is not supported. Resource granularity is coarse as partitions are divided along cell board boundaries. Node Partitions require hardware support which is not provided on smaller servers. As with other physical partitioning systems, the number of partitions which can be created per server is fairly limited.

HP Virtual Partitions

Strengths

Unbound CPUs can be dynamically reallocated between partitions. Virtual Partitions can scale to a large number of CPUs and large quantities of memory (no specific limit in documentation). Different patch levels of HP-UX can be hosted withing different vPartitions. HP Workload Manager was the only workload management system which provided support for monitoring of system *and* application level metrics. This tool can be used to manage Virtual Partitions. Both PA-RISC and Itanium systems are supported by Virtual Partitions.

Virtual Partitions can also be used in conjunction with Node Partitions. A single Node Partition can host multiple (up to 8) Virtual Partitions. Node Partitions can provide hardware fault isolated partitions which can be subdivided into Virtual Partitions to provide finer granularity.

Weaknesses

HP Virtual Partitions can only run HP-UX. All virtual partitions must run the same version of HP-UX (although multiple patch levels are supported). Dynamic reallocation of memory and *bound* CPUs between partitions is not supported. CPU allocation granularity is finer than for Node Partitions, but at a single CPU level is fairly coarse. Memory consumption will be affected by memory overhead of the virtual partition monitor.

HP Integrity Virtual Machines

Strengths

Integrity Virtual Machines supports reasonably fine granularity for allocating resources. A large quantity of memory per VM is supported. This system also supports many partitions per server. Tools are also available to automate reallocation of resources based on system and application goals. Integrity Virtual Machines can be used in conjunction with Node Partitions on Itanium servers to combine the granularity of the system VMM approach with the fault isolation of Node Partitions.

Weaknesses

Integrity Virtual Machines currently support only the HP-UX operating system. The IA64 (Itanium) instruction set architecture is the only one supported by this system. Each partition is limited to a maximum of four CPUs. The virtual machine monitor of this system results in higher memory consumption, limiting scalability to a larger number of partitions on systems with less memory. Reallocating memory or changing the number of CPUs per partition requires a reboot of the affected VMs. Resource isolation for network and disk bandwidth between partitions is absent. Like other software-based partitioning systems Integrity VMs do not provide physical fault isolation.

Fujitsu Physical Partitioning and Fujitsu Extended Partitioning (Xpar)

(These systems are discussed together as they are very similar)

Strengths

Physical partitioning provides isolation from hardware and software failures. Extended Partitions (Xpars) provide the finest CPU granularity of any of the physical partitioning systems evaluated. This is due to the fact that it can provide physical isolation between the two halves of a system board. Dynamic reallocation of both CPUs and memory is another strength of these systems. Scaling to partitions with many CPUs and large quantities of memory is also supported. As with other physical partitioning systems, resource isolation is a strong point as components are dedicated to partitions. Like Sun Dynamic System Domains, Solaris Containers can be used to subdivide partitions providing fine granularity.

Weaknesses

These partitioning systems only support a single ISA (SPARC) and a single operating system (Solaris). Tools to automate resource reallocation based on system goals are not available. As with other physical partitioning systems resource granularity is quite coarse. Fujitsu Physical Partitions allocate all of the CPUs (up to 8) on a system board to when allocating resources. This is coarser than Sun Dynamic System Domains if single core CPUs are used in the Sun systems.

6.4 Evaluation Feasibility

These evaluations demonstrated the feasibility of evaluating virtualisation systems using the framework. The majority of the information was obtained from software and server manuals. A wide variety of information needed to be collected about each system. Collecting this information from a variety of lengthy technical manuals and documents proved challenging, but ultimately successful. Evaluating ten systems is a major undertaking. An evaluation of a single system should be a manageable task for a single person. This is especially true if the person performing the evaluation is familiar with the system being evaluated.

Users wishing to conduct their own evaluations using the framework can use the evaluations documented in this chapter, as an example. The author recommends the references section at the end of this document as a starting point for anyone who wishes to learn more about virtualisation systems.

One of the challenges encountered when evaluating systems using the framework was determining which features are not supported by a given system. Sometimes documentation will clearly state that a given system does not support a particular feature. Other times further investigation was required to determine whether a feature is not supported. The existence of documentation which fails to describe the inclusion of a feature does not necessarily imply that this feature is not supported. Time-consuming searches for documentation were often required to check whether features were supported by systems. White papers from vendors tend to emphasise the strengths of their systems, often failing to describe restrictions and issues. It was more informative to study the manuals available for each system. Relevant information was often provided deep in these manuals.

The framework criteria remained largely unchanged during this chapter. The main changes included defining data types for each of the criteria and clarifying minor points.

6.5 Conclusion

The ten virtualisation systems evaluated in this chapter include most of the major server virtualisation systems available to data centres today. By evaluating a wide range of systems, including three systems not considered in the design phase of the framework, the generality of the framework was demonstrated. The author does not foresee any problems evaluating other virtualisation systems using the framework such as Xen, Linux VServer, Virtuozzo and User Mode Linux (UML) among others.

An understanding of the approaches to virtualisation discussed in Chapter 3 proved valuable in understanding the reasons why certain systems fared better or worse in each category. The classification presented in Chapter 3 was not incorporated into the framework criteria for reasons which were discussed in Chapter 5.

The evaluations presented in this Chapter spanned the majority of the framework criteria, excluding only criteria which cannot be evaluated without purchasing, installing and testing each of the systems. The remaining criteria will be evaluated using Solaris Containers as a case study in the NMMU CoE data centre. This case study is presented in Chapter 7.

One of the stated goals of the framework was to provide enough criteria to be able to highlight the strengths and weaknesses of each system. This application was demonstrated in this chapter. The ranking system presented in Chapter 5 proved useful in identifying the strengths a weaknesses of systems, aiding in the interpretation of the evaluation data. Before this ranking system was adopted, the evaluation data was analysed by inspecting the tables corresponding to each of the evaluation criteria. The rankings were later calculated for each of the systems in each of the categories. There was a strong correlation between the ranking values, and the strengths and weaknesses which were identified by manually inspecting the evaluation data. Some of the strengths and weaknesses only became clear to the author after inspecting the category ranking results. These strengths and weaknesses had gone unnoticed when inspecting the evaluation data manually. The strengths and weaknesses section was later rewritten to include the strengths and weaknesses brought to the author's attention by the rankings. This demonstrated the usefulness of the ranking system and allayed the authors initial concerns about adopting a ranking system for each category. The only anomaly was that Integrity Virtual Machines received a higher than expected ranking in the scalability – scaling to larger partitions category. This was a result of the simplicity of the ranking model. Given the fact that this ranking is intended as an approximation to aid manual interpretation, this was considered acceptable. In all of the other cases the ranking model proved to be accurate. A more accurate ranking system could have been developed, but such a system would have sacrificed the simplicity of the ranking model.

The next chapter describes the selection of Solaris Containers for the NMMU CoE data centre. It also provides an example of a complete evaluation by including criteria which could not be evaluated in this chapter due to practical limitations.

Chapter 7: System Selection and Evaluation Case Study

7.1 Introduction

The practicality of evaluating virtualisation systems using the framework was demonstrated in Chapter 6, based on available literature for practical reasons. This approach is sufficient for most of the framework criteria. For the remaining criteria some experimental work is required. This is especially true for quantitative performance data.

An extended evaluation of Solaris Containers is presented in this chapter as a case study. This evaluation extends beyond literature to include practical aspects. Part of this evaluation was documented in Chapter 6, while the remainder (including practical evaluation) is presented in this chapter. The objectives of this evaluation are to:

- Demonstrate the feasibility of evaluating a system using the practically oriented framework criteria
- Populate the framework with additional information
- Provide an example of a practical evaluation for others to consult, along with the instructions in Chapter 5, when conducting similar evaluations.

The practical evaluation of Solaris Containers is only one aspect of this chapter. One of the applications of the framework identified in Chapter 5 was to facilitate the selection of a suitable virtualisation system. The selection of Solaris Containers for use in the NMMU Telkom Centre of Excellence (CoE) data centre is presented as a case study, demonstrating this application of the framework. This case study is presented first. The selection of Solaris Containers for the CoE data centre is the reason for the evaluation case study being based on this system. This system selection case study serves the following purposes:

- Demonstrate the application of the framework to define requirements
- Demonstrate the application of the framework to facilitate decision-making

This chapter consists of two main sections:

- System selection case study
- System evaluation case study

7.2 System Selection Case Study

One of the applications of the framework is to facilitate the selection of a virtualisation system for use in a data centre, using a virtualisation requirements filter. A filter is used to formally represent a user's requirements for a virtualisation system.

In this section the selection of a virtualisation system for the NMMU CoE data centre, using a filter, is presented as a case study. A brief overview of the data centre is provided next to put this case study in context.

Overview of the NMMU CoE Data Centre

The CoE is currently researching data mining, data warehousing and visualisation of customer network data. This research involves processing large volumes of data using multi-tiered data mining applications.

In order to meet the current and future data processing and research needs of the CoE and department, a small data centre was established in 2004. The author was responsible for the design, implementation and management of this data centre. These details are not relevant to the framework however. This data centre is only relevant to this research as a case study of the selection of a suitable virtualisation system using the framework. Some of the equipment of this data centre was also used to conduct the experimental evaluation of Solaris Containers.

Defining Requirements with a Filter

A virtualisation requirements filter will now be used to define the virtualisation requirements for the NMMU CoE data centre. This serves to demonstrate how a filter can be used to select a suitable virtualisation system for a data centre.

Defining the Filter Production Rules

The rules for a virtualisation requirements filter are defined in order of decreasing importance. The first rule (R1) is always defined as:

R1: add {Sun Dynamic System Domains, HP Node Partitions, Fujitsu Physical Partitions, Fujitsu Extended Partitions, HP Integrity Virtual Machines, HP Virtual Partitions, Solaris Containers, IBM Logical Partitions (pSeries), Microsoft Virtual Server, VMware ESX Server}

The systems in this list represent those which have been evaluated using the framework. When a virtualisation system was selected for the CoE data centre, HP Integrity Virtual machines had not yet been released. The Fujitsu partitioning system had also not yet been evaluated. The candidate set was therefore as follows:

R1: add {Sun Dynamic System Domains, HP Node Partitions, HP Virtual Partitions, Solaris Containers, IBM Logical Partitions (pSeries), Microsoft Virtual Server, VMware ESX Server}

One of the requirements for the data centre was to host multiple tiered applications on the same server. For this reason security and namespace isolation were particularly important. To reflect this the following rules were defined:

```
R2:  if notsame (current.“filesystem isolation”, “Yes”)
     or  notsame (current.“isolation of package databases”, “Yes”)
     or notsame (current.“isolation of network port bindings”, “Yes”)
     or notsame (current.“separate set of users for each partition”, “Yes”)
     or notsame (current.“inter-partition communication isolation”, “Yes”)
     then remove (current) fi
```

Rule R2 represents most of the criteria in the security and namespace isolation category of the framework. Isolation of operating system tuning parameters was not considered essential, as it was unlikely that that type of tuning would be required.

At least six partitions would be required on the same server. With future projects this number is likely to grow. A server with four CPUs was the most suitable given the resource requirements of data mining applications. Even with four CPUs, the number of partitions required would be greater than the number of available CPUs. At least four partitions per CPU would be needed to provide room for future applications. This requirement is represented by the following rule:

R3: **if** aboverange (current.“Maximum partition per CPU”, “4”)
then remove (current) **fi**

Given the resource-intensive nature of tiered data mining applications, each partition would require significant processing resources. In order to take full advantage of the available resources, partitions needed to support at least four CPUs:

R4: **if** aboverange (current.“Maximum CPUs per partition”, “4”)
then remove (current) **fi**

Similarly, some partitions would require at least 1 GB of memory:

R5: **if** aboverange (current.“Maximum memory per partition”, “1 GB”)
then remove (current) **fi**

In order to isolate multiple tiered data mining applications, resource isolation is also important. CPU resource isolation was considered the most important due to the CPU-intensive nature of complex data mining algorithms:

R6: **if** notcontains (current.“CPU resource isolation”, “dedicated”)
and notcontains (current.“CPU resource isolation”, “scheduled”)
then remove (current) **fi**

Due to that fact that this high-availability is not a critical requirement for any of our systems, fault isolation was not considered a major decision factor. High availability is a requirement for commercial data centres, such as those discussed in Chapter 2, but not for this data centre.

The author has some familiarity with the Windows and Solaris operating systems. For this reason the following rule was defined:

R7: **if** notcontains (current.“Operating system compatibility”, “Windows”)
and notcontains (current.“Operating system compatibility”, “Solaris”)
then remove (current) **fi**

In order to maximise the performance/price ratio of the server being purchased at the time, the x86 ISA was selected as the preferred ISA:

R8: **if** notcontains (current.“Instruction set architectures (ISAs) supported”, “x86”)
then remove (current) **fi**

Data warehouses typically store large volumes of data, according to Han and Kamber [HK2001]. Isolating disk bandwidth would therefore be useful. This can be achieved by dedicating disks to partitions (virtual servers), or with scheduling techniques for shared disks:

R9: **if** notcontains (current.“Disk bandwidth isolation”, “dedicated”)
and notcontains (current.“Disk bandwidth isolation”, “scheduled”)
then remove (current) **fi**

Isolation of memory resources was not critical. The applications being hosted were either Oracle databases, with specific memory allocation limits, or Java application server instances, with reasonable maximum heap size settings. Although not critical in this instance, memory resource isolation would provide further protection against memory leaks. Either memory capping or guaranteed allocations would be sufficient. Due to the ordering-based priorities of the rules the following rule is less important than preceding rules:

R10: **if** notcontains (current.“Memory resource isolation”, “guaranteed allocation”)
and notcontains (current.“Memory resource isolation”, “capping”)
then remove (current) **fi**

Applying the Filter Production Rules

In this section, the application of the framework to facilitate decision-making is demonstrated.

First the candidate set is populated using rule R1. This set is as follows:

{Sun Dynamic System Domains, HP Node Partitions, HP Virtual Partitions, Solaris Containers, IBM Logical Partitions (pSeries), Microsoft Virtual Server, VMware ESX Server}

Rule R2 is then applied, but does not result in any systems being eliminated.

Rule R3 eliminates Sun Dynamic System Domains, HP Node Partition and HP Virtual Partitions. This results in the following candidate set:

{Solaris Containers, IBM Logical Partitions (pSeries), Microsoft Virtual Server, VMware ESX Server}

Rule R4 eliminates Microsoft Virtual Server, VMware ESX Server. The candidate set is now reduced to two elements:

{Solaris Containers, IBM Logical Partitions (pSeries)}

Rules R5 and R6 do not result in any of the remaining systems being eliminated.

Rule R7 results in IBM Logical Partitions being eliminated, due to a lack of support for either Solaris or Windows.

As the last remaining system, Solaris Containers has been identified as the most suitable system. There is no need to apply any of the remaining rules. IBM Logical Partitions is the next most suitable system.

Note that in practice rules could be defined using a GUI tool. Users would not have to define rules directly using the filter production rule syntax.

7.3 System Evaluation Case Study

This section presents a case study demonstrating how to evaluate a system using the practically oriented framework criteria, using Solaris Containers on a Sun Fire V40z server as an example.

7.3.1 Test Environment

The practical criteria all require that the system configuration used to perform tests be specified. All tests were performed using a Sun Fire V40z [SM2005f] server. The configuration of this system is described in Table 7.1:

Table 7.1: Test Server Configuration

Processors	Four AMD Opteron Model 848 2.2 GHz
Memory	6 GB (4 x 1GB, 4 x 512 MB)
Disks	3 x 73 GB Ultra320 SCSI 10 000 rpm
Network Adapter	2 x Gigabit Ethernet
Operating System	Solaris 10 (03/05) x86 64 bit full install

For the web server benchmark, a Sun Fire V240 [SM2005g] was used as the test client. The configuration of this server is described in Table 7.2:

Table 7.2: The Configuration of the V240 Test Client

Processor	One UltraSPARC IIIi 1 GHz
Memory	1 GB (4 x 256 MB)
Disks	3 x 73 GB Ultra320 SCSI 10 000 rpm 1 x 36 GB Ultra320 SCSI 10 000 rpm
Network Adapter	4 x Gigabit Ethernet
Operating System	Solaris 9 (09/04)

7.3.2 Evaluation Criteria

Only practically oriented criteria which require practical experimentation are considered here. These criteria are a subset of the criteria selected for each category in Chapter 5.

Manageability (C 3)

Time Taken to Install a Partition (C 3.3)

There are two types of zones which can be created. By default some directories are mounted as read only from the global zone. This type of zone is referred to as a **sparse zone**. Another option is to have no loopback mounts (see Chapter 4). This is referred to as a **full zone**, or full root zone.

Following the instructions provided in Chapter 5, the system was rebooted before testing commenced. Five zones were created, with each zone installed immediately after the previous one's installation was complete. Each of the five zone creations was timed. This test was run once on the V40z server described in table 7.1. First the creation of sparse zones was timed. The system was then reset, and the creation of full zones was timed. The results of these tests are included in Table 7.3 and depicted in Figure 7.1.

Table 7.3 Time Taken to Install a Partition

	Sparse Zone (default)	***Full Zone***
Install time mean (in seconds)	231,8 s	1452 s
Standard deviation	30,3 s	99,48 s

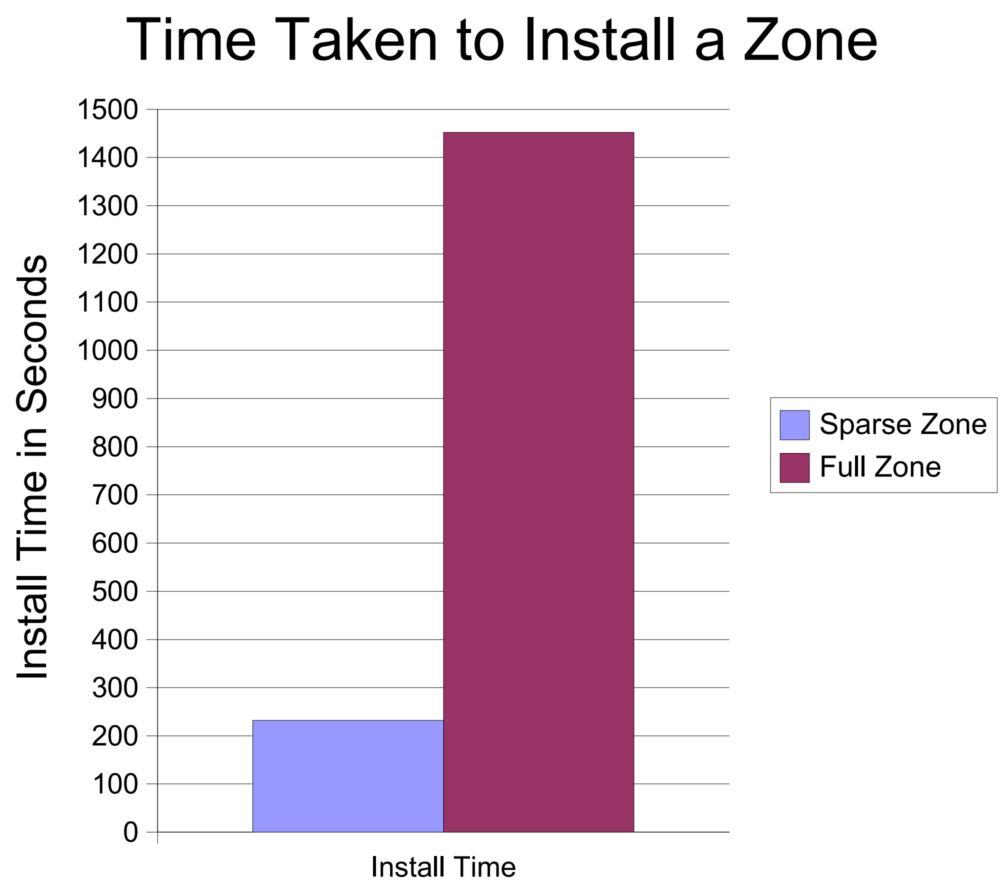


Figure 7.1: Time Taken to Install a Zone

The difference between the time taken to install the different types of zone is large. Full zones took more than six times longer to install. This is clearly illustrated in Figure 7.3. On average, it took just under four minutes to install a sparse zone (the default zone type), and just over 24 minutes to install a full zone.

As with other practically-oriented criteria, the data type for this criterion is a numerical range. As a result, 231,8 seconds is recorded as the lower bound for this criterion, and 1452 seconds is recorded as the upper bound. If other configurations are used to test the installation of zones, the bounds will change.

Granularity (C 5)

CPU Reallocation Time Granularity (C 5.3)

The reallocation time granularity for CPUs was expected to be small, as this is a software-based virtualisation system. Unfortunately, attempts to measure CPU reallocation times were unsuccessful. Every time the command to activate the resource pools functionality was issued, the Solaris kernel would produce a core dump, and reset the system! The reason for this behaviour has yet to be ascertained.

Fortunately, scheduler-based isolation was sufficient for the CoE data centre. As a result, there was never a need to activate this feature, except for experimental purposes.

Not being able to measure the CPU reallocation time granularity of this system was not a major concern. This criterion is only likely to be a differentiating factor between physical partitioning systems.

Memory Reallocation Time Granularity (C 5.4)

Not Applicable (N/A). Dedicating memory to partitions is not currently supported by Solaris Containers.

Scalability – Scaling to More Partitions (C 6.2)

Memory Usage with Fifteen Partitions Booted (C 6.2.4)

Memory usage was measured using Top, a well known Unix utility. This measurement also included any swap space used:

$$\textit{Memory usage} = \textit{real memory} - \textit{free memory} + \textit{swap in use}$$

The V40z server described previously was used for this test. Memory usage was measured after the system had been rebooted. Measurements of memory consumption were taken after each of the zones had finished booting. This experiment was conducted once. After the first three zones, the memory usage increased linearly as the number of zones increased. The change in memory usage after each of the first three zones were booted was slightly higher. Using default settings, booting the first zone resulted in an 88 MB increase in memory usage. This figure decreased to about 67 MB for the last twelve zones booted. The linear increase in memory usage is clearly depicted in Figure 7.2. With fifteen zones booted memory usage stood at 1423 MB.

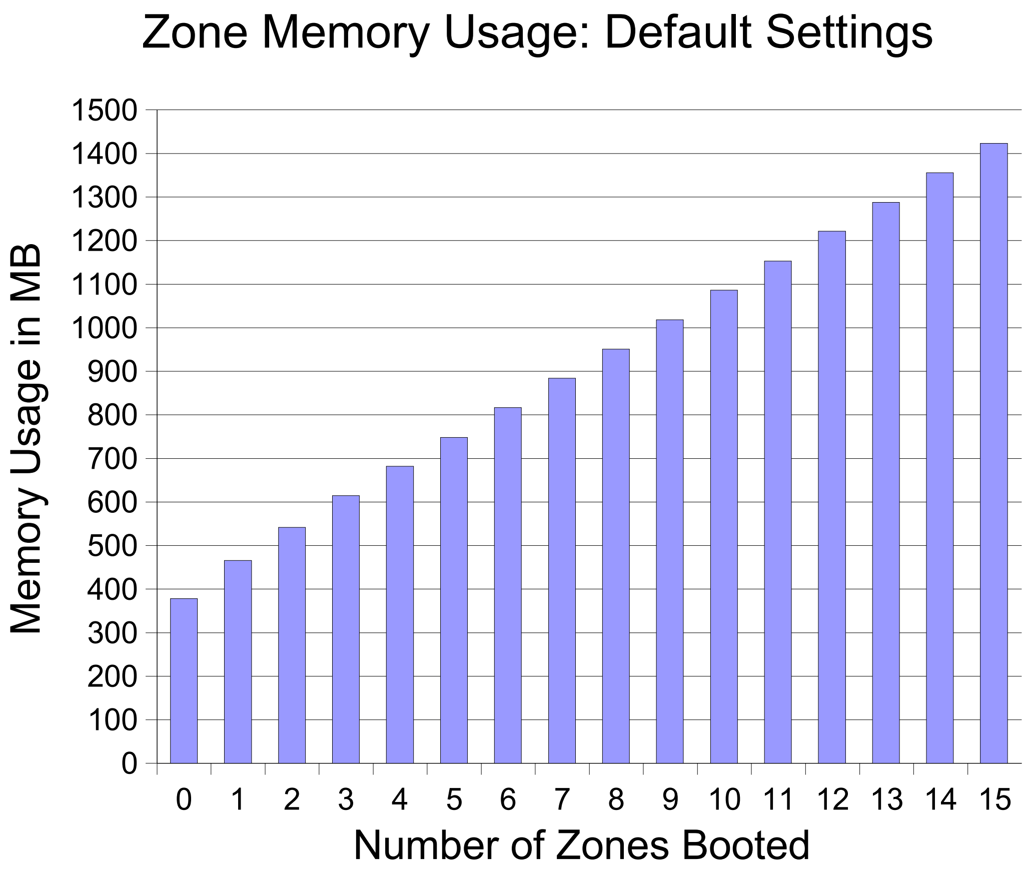


Figure 7.2: Total Memory Usage with Different Numbers of Zones Booted (default settings)

Memory consumption also depends on the configuration of the system. In the memory tests, sparse zones consumed less memory than full zones. This is reflected in Table 7.4 and Figure 7.3. The different zone types were also tested with services disabled. This change resulted in lower memory usage, as can be clearly seen in Figure 7.5. With services disabled, memory usage per sparse zone stabilised at around 48 MB for the last ten zones booted.

Table 7.4: Memory Usage with Fifteen Zones Booted

Configuration	*Memory Usage with Fifteen Zones Booted*
Default settings with services disabled	1060 MB
Full zone with services disabled	1128 MB
Default settings (services enabled)	1423 MB
Full zone with services enabled	1541 MB

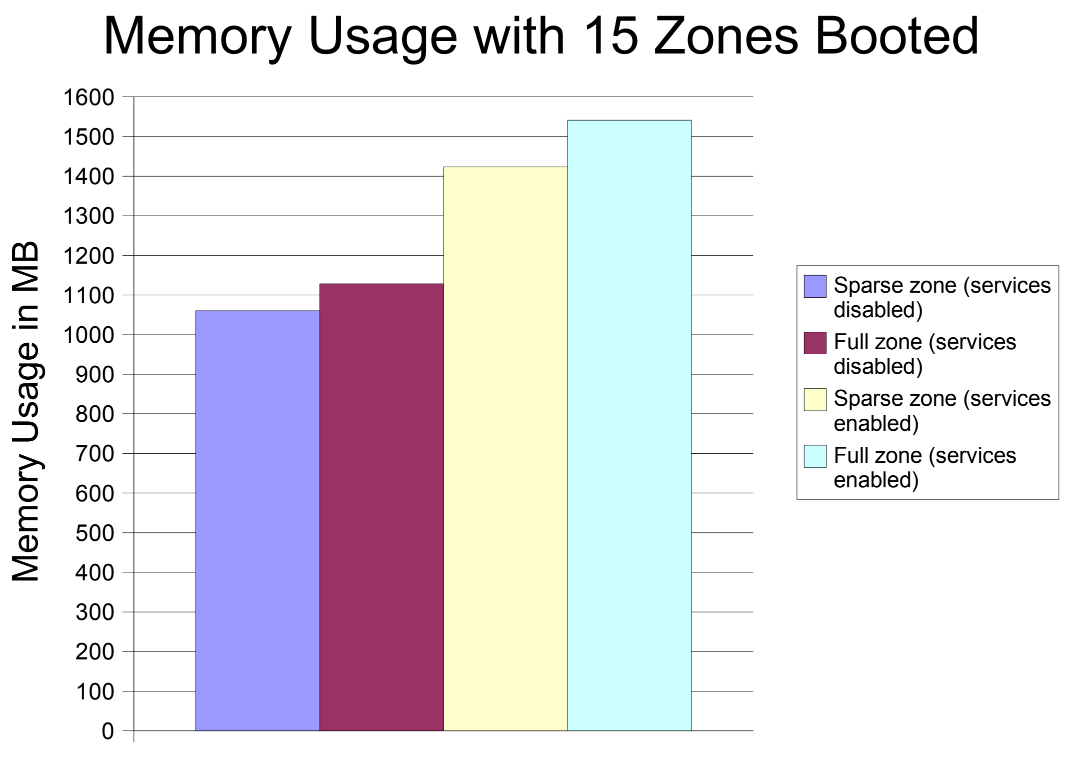


Figure 7.3: Memory Usage With 15 Zones Booted with Different Configurations

The data type for this criterion, as specified in Chapter 5, is a numerical range. 1060 MB forms the lower bound of this range, and 1541 MB the upper bound. If tests are run with other configurations and hardware, these values may change. Nevertheless, all evaluations are included in the framework results.

Disk Space Consumed by a Partition (C 6.2.5)

When Solaris 10 was installed on the V40z, the "full install" option was selected. Disk space consumed by sparse zones and full zones was measured using the following command:

du -ks <zonepath>

A sparse zone and a full zone were created to measure disk the disk space consumed. These measurement need only be taken once. Figure 7.4 depicts the results.

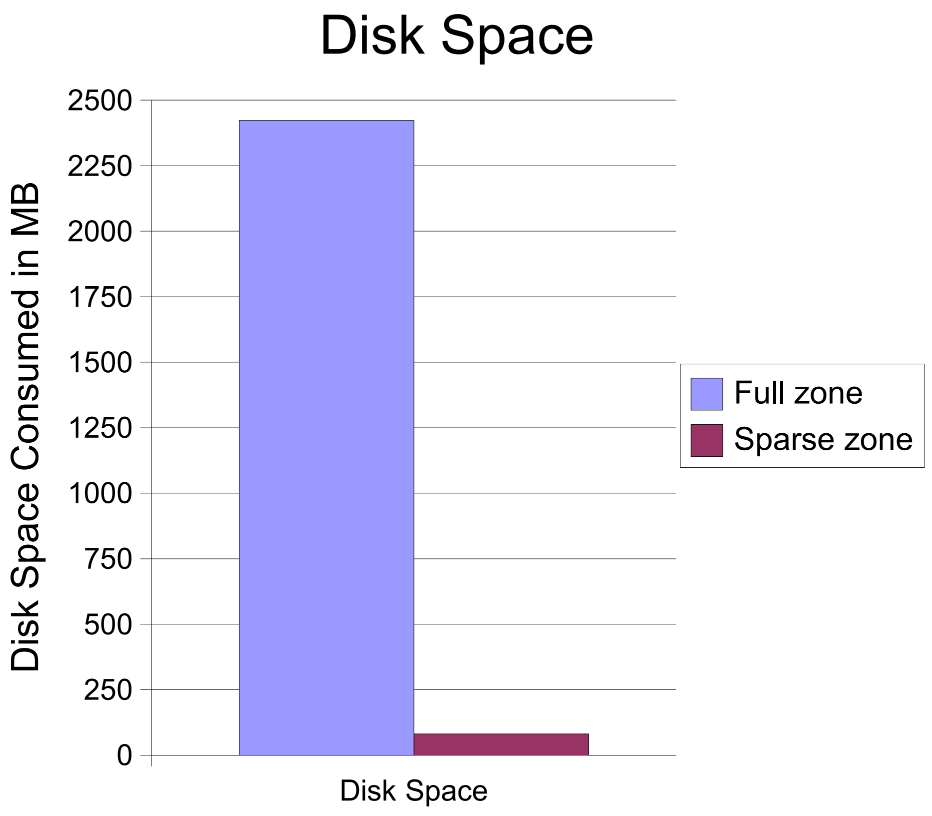


Figure 7.4: Disk Space Consumed by Zones

The sparse zone consumed 81,5 MB and the full zone 2,37 GB. This is due to the larger number of files which are copied to create a full zone. The disk space consumed by a full zone is similar to a complete operating system install.

The result type for this criterion was defined as a numerical range. For this reason 81,5 MB is recorded as the lower bound, and 2,37 GB as the upper bound.

Performance (C 7)

Apache Web Server Benchmark (C 7.1)

Following the instructions provided in Chapter 5 for this criterion, the Apache benchmark was used to measure performance overhead. The V240 server described previously was used as the test client. The V40z server was used as the test server. These two servers were connected directly using a single Gigabit Ethernet connection. The throughput of an Apache 1.3 instance running within a zone was compared to an instance hosted outside a zone. The default zone type, a sparse zone, was used. As per the instructions for this criterion, these measurements were taken six times, with the first measurement discarded as a warm up. The results are summarised in Table 7.5 and depicted in Figure 7.5.

Table 7.5 Apache Throughput for Virtualised and Non-Virtualised Cases

	Zone (virtualised)	***Global (non-virtualised)***
Throughput mean	43056 Kbps	42956,6 Kbps
Virtual/Non-virtual (%)	100,23%	
Difference	100,08 Kbps	
Standard deviation	424,31 Kbps	400,12 Kbps

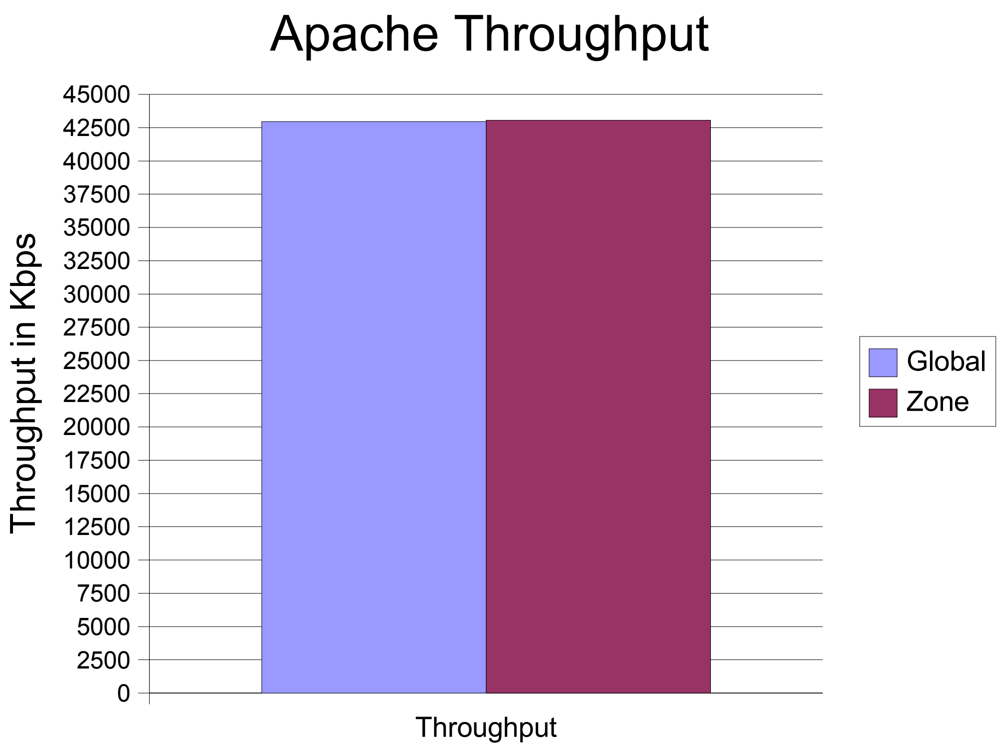


Figure 7.5: Apache Throughput (Zone versus Global)

The difference in performance between the virtualised and non-virtualised cases was smaller than the error margin. This is consistent with the performance claims of Tucker and Comay [TC2004].

The data type for this criterion is specified as a numerical range. As a result, 100,23% is currently both the upper and lower bound for Solaris Containers for this criterion.

Sysbench OLTP Database Benchmark (C 7.2)

As per the instructions provided in Chapter 5 for this criterion, the performance of a MySQL 4.0 database within a zone is compared to the performance when not running within a zone. The V40z server was again used to perform this test. Following the instructions for this criterion, the measurements were taken six times, with the first measurement being discarded as a warm up.

Initially the test failed to run within a zone. Sysbench was attempting to write to a directory mounted from the global zone as read only. The test was being conducted in a sparse zone, as this is the default zone type. This was a clear example illustrating the importance of filesystem isolation (criterion C 2.3.1 in Chapter 5). Sparse zones do not satisfy the filesystem isolation requirements defined for this criterion, but full zones do. For this reason these tests had to be conducted using a full zone.

As stated in Chapter 5 this test is to be run with the default settings. This is achieved with the following two commands:

```
./sysbench --mysql-user=root --test=oltp prepare
./sysbench --mysql-user=root --test=oltp run
```

Table 7.6 summarises the results using this benchmark. Figure 7.6 provides a graphical representation.

Table 7.6: Sysbench OLTP Transactions Per Second (tps)

	Zone (virtualised)	***Global (non-virtualised)***
Throughput mean	120,83 tps	122,81 tps
Virtual/Non-virtual (%)	98,39%	
Difference	1,97 tps	
Standard deviation	2,44 tps	1,44 tps

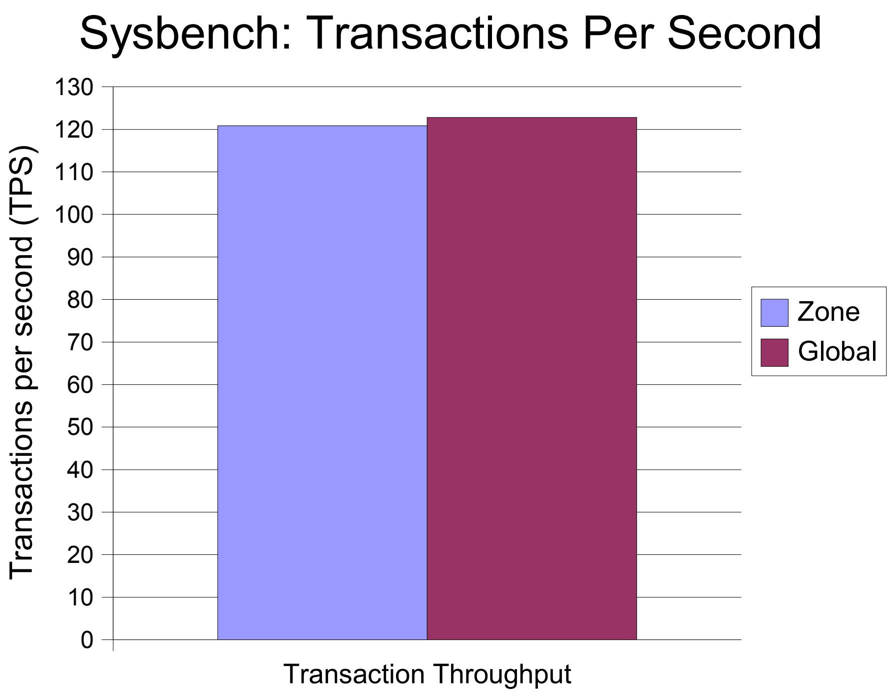


Figure 7.6: Sysbench OLTP Throughput

The throughput of the virtualised case was very close to the non-virtualised case. Once again, the difference was smaller than the standard deviation. This result implies that the performance overhead was too small to measure accurately.

The result type for this criterion is a numerical range. For this reason, a value of 98,38 % is recorded as the upper and lower bound for this criterion.

7.4 Conclusion

One of the applications of the framework is to facilitate the selection of virtualisation systems. Defining requirements for the NMMU CoE data centre using a virtualisation requirements filter proved to be a straightforward task. This case study demonstrated the ease with which the framework can be used to facilitate decision-making. Solaris Containers was identified as the most suitable system for the data centre, using the framework. This system was successfully deployed in the CoE data centre, validating the selection. Solaris Containers proved to be very useful when managing the data centre. Oracle upgrades were simplified. Application failures and configuration errors were isolated. This proved to be useful on more than one occasion. Root privileges could be delegated on a per-container basis. Installing multiple instances of applications was also greatly simplified, not least of which because of isolation of network port bindings. This system also provided the scalability necessary to support both large partitions, for demanding data warehousing applications, and a sufficient number of partitions to support multiple projects and application tiers. These and other benefits correspond to many of the advantages of virtualisation identified in Chapter 3.

The low overhead of Solaris Containers, which was claimed by Tucker and Comay [TC2004], was confirmed experimentally, using the practically oriented framework criteria. It will be interesting to note the effect of the upcoming ZFS file system [SM2005h] for Solaris, on zone installation times, and disk space usage. The copy-on-write approach of this file system is likely to greatly reduce disk space usage of zones, and decrease zone installation times. Once this system is released officially, these tests will be rerun.

This evaluation case study demonstrated the feasibility of conducting evaluations using these criteria. The conciseness of the practically oriented framework criteria ensured that the evaluation process was reasonably simple to perform. This process was also simplified by virtue of the fact that the framework performance criteria specify the use of default tuning and benchmark settings wherever possible. Anyone wishing to perform a similar evaluation, using the framework, could use the case study presented in this chapter as a practical example.

Chapter 8: Conclusions

8.1 Introduction

The main objective of this research was to develop a framework for conducting comparative evaluations of server virtualisation systems. The purpose of this chapter is to reflect on this framework. The conclusions which can be drawn from this work, future research, and the significance of this work are also discussed.

This chapter covers the following topics:

- Evaluation of Outcomes
- Significance of the Research
- Future Research

8.2 Evaluation of Outcomes

The objectives of this research were outlined in Chapter 1. These objectives will now be revisited to evaluate the extent to which they were achieved. The four main objectives which were identified are discussed first. This is followed by a discussion of the framework design principles.

Objectives

The first objective was to develop a framework for conducting comparative evaluations of server virtualisation systems. This was successfully done. The feasibility of conducting comparative evaluations using this framework was demonstrated by evaluating ten virtualisation systems.

The second objective was to facilitate the identification of system strengths and weaknesses using the framework. This application was clearly demonstrated. The strengths and weaknesses of ten major virtualisation systems were identified using the framework. The ranking system which was developed also proved useful in facilitating the identification of these strengths and weaknesses.

The third objective was to provide a means to define requirements based on the framework criteria in order to facilitate system selection. The virtualisation requirements filter mechanism proposed satisfies this objective. Decision-makers can define requirements in terms of the framework using such a filter. This filter can be applied to identify a suitable virtualisation system matching these requirements. The NMMU Telkom CoE data centre was used as a case study to demonstrate this application. A suitable virtualisation system was successfully identified for use in this data centre, using a requirements filter. The system which was selected, Solaris Containers, proved to be a good choice, meeting the needs of the CoE as described in Section 7.4.

The fourth objective was to gather information about current virtualisation systems, in the form of evaluations using the framework. This objective was clearly satisfied by the evaluation of ten major virtualisation systems using the framework. These evaluations were used to populate the framework with information about the major commercial virtualisation systems available today.

Framework Design Principles

In addition to the main research objectives, six framework design principles were identified in Chapter 1:

- **Objective:**
  The first design principle was that the framework should be objective. Throughout the design of the framework this principle was adhered to. The objectivity of the framework was demonstrated by successfully identifying the major weaknesses of all of the ten virtualisation systems evaluated. During the design phase, benchmarks with any connection to virtualisation systems vendors were excluded. Freely available, open source benchmarks were selected to ensure transparency.

- **Thorough**

  The framework criteria were sufficient to identify the main strengths and weaknesses of all of the ten virtualisation systems evaluated. This demonstrated the thoroughness of the 40 framework criteria. There is still room to add additional framework criteria. This is especially true for the manageability and performance sections, but this would result in an overly complex framework.

- **Balance Between Thoroughness and Conciseness**

  Despite the complexity of the problem domain, a balance has been struck between complexity and thoroughness. Although 40 criteria is more than was originally envisioned, this was necessary to satisfy the thoroughness requirement. With seven main categories, and three additional subcategories, there are an average of four criteria per framework subcategory. This is a reasonable balance. Including additional criteria, especially in the performance category, would greatly increase the effort required to evaluate a system and interpret the results. By restricting the number of framework criteria, evaluating a single system is still a reasonable task.

- **Generic**

  The ten virtualisation systems which were evaluated using the framework included examples of all of the different types of virtualisation systems. This demonstrated the generality of the framework criteria. Avoiding the inclusion of system-specific implementation details, such as optimisations employed, in the framework helped to ensure this.

- **Extensible**

  The generality of the framework categories will make the inclusion of additional criteria reasonably straightforward. Each category, e.g. compatibility, is general enough to easily be extended with additional criteria. The inclusion of additional framework categories is also possible, by adding additional nodes to the tree representing the framework.

- **Relevance to Data Centres**
  The applications of virtualisation in the data centre which were identified were taken into account when the framework criteria were selected. Due to practical constraints, this was based on literature. A more thorough approach would have been to consult managers of major data centres, preferably in different countries and across different industries. Unfortunately, such an investigation was not practical.

8.3 Significance of the Research

This framework is of practical benefit to researchers, practitioners and vendors.

Researchers who are new to the field can consult the evaluations conducted using the framework to learn about current virtualisation systems in days or weeks, rather than months or years. This framework is a knowledge structure, bringing order to information about virtualisation systems. Information about these systems is currently unstructured, voluminous and dispersed among multiple sources. Using the framework, researchers can rapidly get a much clearer picture of the current state of the art in server virtualisation. The literature study of data centres, virtualisation and current virtualisation systems is particularly valuable, bringing together information from a wide range of sources. The references section at the end of this dissertation is also a rich source of information.

Practitioners, such as data centre managers and system administrators also stand to benefit from this research. In addition to providing a structured source of detailed information about server virtualisation systems, the framework can also be used to facilitate system selection. Virtualisation requirements for a data centre can be defined in terms of the framework criteria. This information can be used to identify the systems which are best suited to a particular environment. The theoretical foundations laid by this research could be used as the foundation for a decision support system.

As the use of virtualisation systems becomes commonplace, users lacking an understanding of the principles of virtualisation will increasingly be exposed to such systems. If these users had a framework to consult, their understanding of virtualisation and current virtualisation systems could be greatly improved. These users would then be able to make more informed decisions when selecting these systems.

Vendors and researchers implementing virtualisation systems could use the framework to rapidly identify the relative strengths and weaknesses of current virtualisation systems. This information is particularly useful to developers of commercial virtualisation systems. These developers may wish to learn about the features provided by competing systems in order to identify areas for improvement. Practitioners, researchers and vendors may be knowledgeable in a certain virtualisation system, but may wish to learn more about other systems. Information about strengths and weaknesses is also useful to decision-makers, and anyone wishing to learn more about virtualisation systems. The ranking system, which was developed during the course of this research, can be used as a quick reference for identifying the relative strengths and weaknesses of systems.

The ten current virtualisation systems which have been evaluated using the framework include virtually all of the major commercial virtualisation systems available today. Information provided by evaluations is structured and rich in content.

The framework which is the subject of this research is believed to the only one of its kind in existence.

8.4 Future Research

One opportunity to extend this work would be to evaluate additional systems using the framework. Examples of systems which have yet to be evaluated include Virtuozzo [Vir2005], Linux VServer [LVS2005, BL2005], User Mode Linux (UML) [Dik2000] and Xen [BDF+2003], among others. Xen in particular is generating a lot of interest. Originally an academic research project, this system is currently being commercialised.

The virtualisation requirements filter presented in Chapter 5 could be used as a theoretical foundation for the development of a decision support system. This system could provide a graphical interface, making is easy for practitioners such as data centre managers to define requirements for new systems, and compare available systems.

Another area for future research is the refinement of the ranking model presented in Chapter 5. This ranking system is reasonably simple, but there is room for improvement, as noted in Chapter 6. Any refinements to this model should be balanced against the need to keep this ranking system reasonably easy to understand.

A logical extension of this work would be to create a publicly-available, centralised body of knowledge about the capabilities of different server virtualisation systems, structured around the framework. A website to publish the results of evaluations and information about the framework could provide access to this information. This website could enable the author to receive feedback from researchers in the field, and practitioners, in order to further refine the framework. Given the framework criteria, researchers and vendors could submit detailed evaluations of virtualisation systems in order to fully populate this body of knowledge. All evaluations submitted would be reviewed by the author, to ensure consistency and accuracy. Evaluations would be published on-line subject to public scrutiny to further eliminate any inaccuracies. This body of knowledge could be used to stimulate debate around, and interest in, virtualisation.

References

[AAR2002] Andrzejak, A., Arlitt, M., and Rolia, J., Bounding the Resource Savings of Utility Computing Models, HP Laboratories Palo Alto technical report HPL-2002-339, 2002

[ABC+1966] Adair, R., Bayles, R. U., Comeau, L. W., and Creasy, R. J., A Virtual Machine System for the 360/40, IBM Cambridge Scientific Center Report No G320-2007, 1966

[AL2000] Abdelzaher, T. F., and Lu, C., Modeling and Performance Control of Internet Servers, 39th IEEE Conference on Decision and Control, 2000

[AG2001] Alicherry, M., Gopinath, K., Predictable Management of System Resources for Linux, Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, pages 273-283, 2001

[AID2002] Aron, M., Iyer, S., Druschel, P., A Resource Management Framework for Predictable Quality of Service in Web Servers, submitted for publication, circa 2002

[AMD2001] Advanced Micro Devices, Inc., AMD 64-Bit Technology: The AMD x86-64 Architecture Programmers Overview, http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/x86-64_overview.pdf, 2001, Retrieved: 2005

[AMD2005] Advanced Micro Devices, Inc., AMD “Pacifica” Virtualization Technology, http://enterprise.amd.com/downloadables/Pacifica.ppt, Retrieved: 2005

[AP2004] Arregoces, M., Portolani, M., Data Center Fundamentals, First Edition, Cisco Press, Indianapolis USA, 2004

[AS2004] Singh, A., An Introduction to Virtualization, http://www.kernelthread.com/publications/virtualization, Retrieved: 2004

[ASF2005a] The Apache Software Foundation, Apache HTTP Server Project, http://httpd.apache.org/, Retrieved: 2005

[BEA2005] BEA Systems, WebLogic Server 9.0, http://commerce.bea.com/showproduct.jsp?family=WLS& major=9.0&minor=0, Retrieved: 2005

[BCG1973] Buzen, J. P., Chen, P. P., Goldberg, R. P., A Note on Virtual Machines and Software Reliability, Summary of a paper from IEEE Symposium on Software Reliability, 1973

[BDF+2003] Barham, P, Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugbauer, R., Pratt, I., Warfield, A., Xen and the Art of Virtualization, Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 164 - 177, 2003

[BDM1999] Banga, G., Druschel, P., and Mogul, J. C., Resource Containers: A New Facility for Resource Management in Server Systems, 3rd Symposium on Operating Systems Design and Implementation, 1999

[BE2004] Boutcher, D., and Engebretsen, D., Linux Virtualization on IBM POWER5 Systems, Ottawa Linux Symposium, 2004

[BH1975] Belpaire, G., and Hsu, N., Formal Properties of Recursive Virtual Machine Architectures, SOSP '75: Proceedings of the Fifth ACM Symposium on Operating Systems Principles, pages 89-96, 1975

[BFH+1975] Bagley, J. D., Floto, E. R., Hsieh, S. C., and Watson, V. , Sharing Data and Services in a Virtual Machine System, SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles, pages 82 – 88, 1975

[BGO+1998] Bruno, J., Gabber, E., Ozden, B., and Silberschatz, A., The Eclipse Operating System: Providing Quality of Service via Reservation Domains, Proceedings of the USENIX Annual Technical Conference (NO 98), 1998

[BH1973] Bellino, J, Hans, C., Virtual Machine or Virtual Operating System, Proceedings of the ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems pages 20-29, 1973

[BL2005] des Ligneris, B. Virtualization of Linux Based Computers: The Linux-VServer Project, Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS'05), pages 340-346, 2005

[CFH+2005] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., Warfield, A., Live Migration of Virtual Machines, Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, 2005

[CGS2003] Chandra, A., Goyal, P., and Shenoy, P., Quantifying the Benefits of Resource Multiplexing in On-Demand Data Centers, First ACM Workshop on Algorithms and Architectures for Self-Managing Systems (Self-Manage 2003), 2003

[CN2004] CNET News, Microsoft Ready to Ship Virtual Server http://news.com.com/Microsoft+ready+to+ship+Virtual+Server/2100-1016_3-5360520.html?tag=nl, 2004, Retrieved: 2005

[Cre1981] Creasy, R. J., The Origin of the VM/370 Time-Sharing System, IBM Journal of Research and Development vol. 25, no. 5, pages 483-490, 1981

[CRS+2004] Carolan, J., Radeztsky, S., Strong, P., Turner, E., Building N1 Grid Solutions - Preparing, Architecting, and Implementing Service-Centric Data Centers, Prentice Hall and Sun Microsystems Press, United States of America, 2004

[CS2002a] Cisco Systems, Inc., Enterprise Data Center Design, www.cisco.com/application/pdf/en/us/guest/netsol/ns304/c649/ccmigration_09186a00800d67f5.pdf, 2002, Retrieved: 2004

[CS2003a] Cisco Systems, Inc., Data Center Networking Solutions, http://www.cisco.com/application/pdf/en/us/guest/netsol/ns304/c654/cdccont_0900aecd80096182.pdf, 2003, Retrieved: 2004

[Den2001] Denning, P. J., Anecdotes, IEEE Annals of the History of Computing issue 1058-6180/00 page 73, 2001

[Dik2000] Dike, J., A User-Mode Port of the Linux Kernel, Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, Georgia USA, 2000

[Dik2001a] Dike, J., User Mode Linux - Running Linux on Linux, Linux Magazine April 2001 issue

[Dik2001b] Dike, J., User-Mode Linux, Ottawa Linux Symposium 2001

[EG2004] The Edison Group (prepared for Sun Microsystems), Sun Solaris Container Technology vs. IBM Logical Partitions and HP Virtual Partitions, http://www.sun.com/datacenter/consolidation/docs/Sun_Solaris_Container_vs_IBM_HP_Partition.pdf, 2004, Retrieved: 2005

[FCC1999] Federal CIO Council, Federal Enterprise Architecture Framework, https://secure.cio.noaa.gov/hpcc/docita/files/federal_enterprise_arch_framework.pdf, 1999, Retrieved: 2005

[FDF2005] Figueiredo, R., Dinda, P. A., and Fortes, J., Resource Virtualization Renaissance, IEEE Computer, May 2005, pages 28–31

[FHN+2004] Fraser, K., Hand, S, Neugebauer, R., Pratt, I, Warfield, A., and Williamson, M., Safe Hardware Access with the Xen Virtual Machine Monitor, 1st Workshop on Operating System and Architectural Support for the on demand IT Infrastructure (OASIS'04), 2004

[Fir2002] Firesmith, D., Data Center, http://www.donald-firesmith.com/Components/WorkProducts/ImplementationSet/DataCenter/DataCenter.html, 2002, Retrieved: 2005

[FS2003a] Fujitsu Siemens Computers GmbH, Partition Operation Guide, http://manuals.fujitsu-siemens.com/infothek/sol_os_en/books/partopguide05en/partopguide05en.pdf, 2003, Retrieved: 2005

[FS2004a] Fujitsu Siemens Computers GmbH, Dynamic Reconfiguration: Introduction to the Procedure, http://manuals.fujitsu-siemens.com/infothek/sol_os_en/books/dynreconf_e2/dynreconf_e2.pdf, 2004, Retrieved: 2005

[FS2004b] Fujitsu Siemens Computers GmbH, PrimePower Manageability, http://www.fujitsu-siemens.com/resources/116/10770870.pdf, 2004, Retrieved: 2005

[FS2005a] Fujitsu Siemens Computers GmbH, PrimePower 2500 Manual, http://manuals.fujitsu-siemens.com/primpower/ UM_2500v4_fsc_us.pdf, August 2005 Edition, Retrieved: 2005

[FS2005b] Fujitsu Siemens Computers GmbH, PrimePower Enterprise Servers, http://facts.fujitsu-siemens.com/index.cfm?fuseaction=dspcontent& lid=130, Retrieved: 2005

[GdK2004] de Kock, G. D. V., Introduction to Artificial Intelligence – Expert Systems, University of Port Elizabeth, fifth edition, 2004

[Gol1972] Goldberg, R. P., Architectural Principles for Virtual Computer Systems, Ph.D. thesis, Harvard University, 1972

[Gol1973] Goldberg, R. P., Architecture of Virtual Machines, Proceedings of the AFIPS National Computer Conference, Second Edition pages 74-112, 1973

[Gum1983] Gum, P. H., IBM Research and Development Journal, Volume 27, No. 6, pages 530–544, November 1983

[HBS2002] Höxer, H., Buchacker, K., Sieh, V., Implementing a User-Mode Linux with Minimal Changes from Original Kernel, 9th International Linux System Technology Conference, pages 72-82, 2002

[HK2001] Han, J., Kamber, M., Data Mining – Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, 2001

[Kot2001] Kotov, V, On Virtual Data Centers and Their Operating Environments, http://www.hpl.hp.com/techreports/2001/HPL-2001-44.pdf, 2001, Retrieved: 2005

[HP2002] Hewlett Packard Development Company, L.P., Partitioning - usage in the real world with superdome and rp8400 white paper, http://www.hp.com/products1/unix/operating/manageability/ partitions/library/partitioningcasestudy.pdf, 2002, Retrieved: 2004

[HP2003a] Hewlett Packard Development Company, L.P., HP-UX virtual partitions (vPars) white paper, http://www.hp.com/products1/unix/operating/manageability/partitions/library/vpars_wp203.pdf, 2003, Retrieved: 2004

[HP2004a] Hewlett-Packard Development Company, L.P., System Architecture, http://www.hpl.hp.com/research/dca/system/, Retrieved: 2004

[HP2004b] Hewlett Packard Development Company, L.P., HP Partitioning Continuum for HP-UX11i on HP 9000 and HP Integrity servers, http://h71028.www7.hp.com/ERC/downloads/5982-9141EN.pdf, Retrieved: 2004

[HP2004c] Hewlett Packard Development Company, L.P., HP System Partitions Guide - Administration for nPartitions, Tenth Edition, http://docs.hp.com/hpux/pdf/5990-8170.pdf, Retrieved: 2004

[HP2004d] Hewlett Packard Development Company, L.P., HP-UX Workload Manager Overview, http://www.hp.com/products1/unix/operating/docs/wlm.overview.pdf, Retrieved: 2004

[HP2004e] Hewlett Packard Development Company, L.P., Hewlett Packard (HP), http://www.hp.com, Retrieved: 2004

[HP2004f] Hewlett Packard Development Company, L.P., HP-UX Virtual Partitions Ordering and Configuration Guidelines, http://docs.hp.com/en/1705/oc.pdf, Retrieved: 2004

[HP2004g] Hewlett Packard Development Company, L.P., Using HP-UX Workload Manager effectively with your most critical applications, http://www.hp.com/products1/unix/operating/docs/wlm.toolkits.pdf, Retrieved: 2004

[HP2004h] Hewlett Packard Development Company, L.P., Workload management for HP-UX, Retrieved: 2004

[HP2004i] Hewlett Packard Development Company, L.P., nPartitions, http://www.hp.com/products1/unix/operating/manageability/partitions/nPartitions.html, Retrieved: 2004

[HP2004j] Hewlett Packard Development Company, L.P., HP-UX Virtual Partitions (vPars) T1335AC : Ordering and Configuration Guidelines, http://docs.hp.com/hpux/onlinedocs/1705/oc.pdf, Retrieved: 2004

[HP2004k] Hewlett Packard Development Company, L.P., Installing and Managing HP-UX Virtual Partitions (vPars) - Fourth Edition, http://docs.hp.com/hpux/onlinedocs/T1335-90027/vparsguide.90027.pdf, Retrieved: 2004

[HP2005a] Hewlett Packard Development Company, L.P., Configurable Tuning Parameters, http://docs.hp.com/en/939/KCParms/KCparams.OverviewAll.html, Retrieved: 2005

[HP2005b] Hewlett Packard Development Company, L.P., PA-RISC 2.0 instruction set architecture, http://h21007.www2.hp.com/dspp/tech/tech_TechDocumentDetailPage_IDX/1,1701,959,00.html, Retrieved: 2005

[HP2005c] Hewlett Packard Development Company, L.P., HP-UX 11i, http://www.hp.com/products1/unix/operating/index.html, 2005

[HP2005d] Hewlett Packard Development Company, L.P., HP 9000 Superdome Server, http://www.hp.com/products1/servers/scalableservers/superdome/, Retrieved: 2005

[HP2005e] Hewlett Packard Development Company, L.P., HP Integrity high-end servers, http://www.hp.com/products1/servers/integrity/superdome_high_end/comparison.html, Retrieved: 2005

[HP2005f] Hewlett Packard Development Company, L.P., HP Global Workload Manager—Improving server CPU utilization technical overview http://h71028.www7.hp.com/ERC/downloads/5983-0505EN.pdf, Retrieved: 2005

[HP2005g] Hewlett Packard Development Company, L.P., HP Integrity Virtual Machines Installation, Configuration, and Administration, http://docs.hp.com/en/T2767-90004/installing.pdf, Retrieved: 2005

[HP2005h] Hewlett Packard Development Company, L.P., Installing and Managing HP-UX Virtual Partitions (vPars) - Sixth Edition Update http://docs.hp.com/en/6769/T1335-90038.pdf, Retrieved: 2005

[HP2005i] Hewlett Packard Development Company, L.P., HP System Partitions Guide – Administration for nPartitions, Twelfth Edition, http://docs.hp.com/en/5991-1237/5991-1237.pdf, Retrieved: 2005

[HP2005j] Hewlett Packard Development Company, L.P., HP Integrity Superdome Servers: QuickSpecs, http://h18000.www1.hp.com/products/quickspecs/11717_div/11717_div.PDF, Retrieved: 2005

[HP2005k] Hewlett Packard Development Company, L.P., HP Partition Manager, http://docs.hp.com/en/PARMGR2/, Retrieved: 2005

[HP2005l] Hewlett Packard Development Company, L.P., Global Workload Manager, http://h71028.www7.hp.com/enterprise/cache/257279-0-0-0-121.aspx, Retrieved: 2005

[IBM2002a] International Business Machines Corporation, LPAR For Decision Makers, www-1.ibm.com/servers/eserver/pseries/hardware/white papers/lpar_decision.pdf, 2002, Retrieved: 2004

[IBM2002b] International Business Machines Corporation, LPAR Configuration and Management Working with IBM iSeries Logical Partitions, http://www.redbooks.ibm.com/redbooks/pdfs/sg246251.pdf, 2002, Retrieved: 2004

[IBM2003a] International Business Machines Corporation, The Complete Partitioning Guide for IBM eServer pSeries Servers, www.redbooks.ibm.com/redbooks/pdfs/sg247039.pdf, 2003, Retrieved: 2004

[IBM2003b] International Business Machines Corporation, IBM eServer pSeries Dynamic Logical Partitioning (DLPAR) in AIX 5.2 with Oracle, http://www.redbooks.ibm.com/redpapers/pdfs/redp3735.pdf, 2003, Retrieved: 2004

[IBM2003c] International Business Machines Corporation, Introduction to Storage Area Networks, http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf, 2003, Retrieved: 2004

[IBM2004a] International Business Machines Corporation, Advanced POWER Virtualization on IBM eServer p5 Servers: Introduction and Basic Configuration, http://www.redbooks.ibm.com/redbooks/pdfs/sg247940.pdf, Retrieved: 2004

[IBM2004b] International Business Machines Corporation (IBM), http://www.ibm.com, Retrieved: 2004

[IBM2004c] International Business Machines Corporation, Virtualization and the On Demand Business, www.redbooks.ibm.com/redpapers/pdfs/redp9115.pdf, Retrieved: 2004

[IBM2004d] International Business Machines Corporation, Mainframe servers: zSeries, www.ibm.com/servers/eserver/zseries/, Retrieved: 2004

[IBM2004e] International Business Machines Corporation, Unix servers: eServer p5 and pSeries, www.ibm.com/servers/eserver/pseries/, Retrieved: 2004

[IBM2004f] International Business Machines Corporation, Advanced POWER Virtualization on IBM eServer p5 Servers, http://www.redbooks.ibm.com/redpieces/pdfs/sg245768.pdf, Retrieved: 2004

[IBM2004g] International Business Machines Corporation, Partitioning for AIX fourth edition, http://publib.boulder.ibm.com/infocenter/eserver/v1r2s/en_US/info/iphbk/iphbk.pdf, Retrieved: 2004

[IBM2004h] International Business Machines Corporation, Logical Partitions on IBM PowerPC: A guide to working with LPAR on Power5 i5 servers, http://www.redbooks.ibm.com/redpieces/pdfs/sg248000.pdf, Retrieved: 2004

[IBM2005a] International Business Machines Corporation, IBM eServer OpenPower 720: Server Consolidation Using Advanced OpenPower Virtualization (White Paper), http://www-1.ibm.com/servers/eserver/openpower/ white papers/720servconsol_apov.pdf, Retrieved: 2005

[IBM2005b] International Business Machines Corporation, Partitioning Implementations for IBM eServer p5 Servers, http://www.redbooks.ibm.com/redbooks/pdfs/sg247039.pdf, Retrieved: 2005

[IBM2005c] International Business Machines Corporation, Server Consolidation with VMware ESX Server, http://www.redbooks.ibm.com/ redpapers/pdfs/redp3939.pdf, Retrieved: 2005

[IBM2005d] International Business Machines Corporation, Power Architecture , http://www-03.ibm.com/chips/power/index.html, Retrieved: 2005

[IBM2005e] International Business Machines Corporation, IBM AIX 5L, http://www-03.ibm.com/servers/aix/, Retrieved: 2005

[IBM2005f] International Business Machines Corporation, IBM eServer p5 595, http://www-03.ibm.com/servers/eserver/pseries/hardware/highend/ 595.html, Retrieved: 2005

[IBM2005g] International Business Machines Corporation, DB2 Product Family, http://www-306.ibm.com/software/data/db2/, Retrieved: 2005

[IBM2005h] International Business Machines Corporation, IBM LPAR Validation Tool, http://www-03.ibm.com/servers/eserver/iseries/lpar/systemdesign.html, Retrieved: 2005

[IBM2005i] International Business Machines Corporation, WebLogic Application Server, http://www-306.ibm.com/software/websphere/sw-bycategory/subcategory/SW620.html, Retrieved: 2005

[Int2004a] Intel Corporation, Storage Area Network (SAN), ftp://download.intel.com/design/network/solutions/manual/ Chapter9.pdf, Retrieved: 2004

[IN2004] Internet News, IBM Revs Up its 'Virtualization Engine', http://www.internetnews.com/ent-news/article.php/3346341, Retrieved: 2004

[JXE2004] Jiang, X., Xu, D., Eigenmann, R., Protection Mechanisms for Application Service Hosting Platforms, Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid, 2004

[KC2002] King, S. T., and Chen, P. M., Operating System Extension to Support Host Based Virtual Machines, University of Michigan technical report cse-tr-465-02, http://www.eecs.umich.edu/techreports/cse/2002/CSE-TR-465-02.pdf, 2002, Retrieved: 2005

[KC2003] Kakadia, D, Croucher, R., - Sun Microsystems, Network Design Patterns: N-Tier Data Centres, http://www.sun.com/blueprints/1203/817-4683.pdf, 2003, Retrieved: 2004

[KCL+1998] Keil, M, Cule, P.E., Lyytinen, K., and Schmidt, R.C., A Framework for Identifying Software Project Risk, Communications of the ACM Vol. 41, No. 11, 1998

[KDC2003] King, S. T., Dunlap, G. W., Chen, P. M., Operating System Support for Virtual Machines, Proceeding of the 2003 USENIX Technical Conference, 2003

[KW2000] Kamp, P., and Watson, R. N. M., Jails: Confining the Omnipotent Root, 2nd International System Administration and Networking Conference, 2000

[LD1998] Loosley, C., Douglas, F., High-Performance Client/Server, A Guide to Building and Managing Robust Distributed Systems, John Wiley & Sons, Inc. New York, 1998

[lmb1996] McVoy, L., Staelin, C., lmbench: Portable Tools for Performance Analysis, Proceedings of the USENIX 1996 Annual Technical Conference, 1996

[LVS2005] Linux-VServer Project, http://www.linux-vserver.org, Retrieved: 2005

[MS2004a] Microsoft Corporation, Virtual Server 2005 Frequently Asked Questions, http://www.microsoft.com/windowsserversystem/virtualserver, Retrieved: 2004

[MS2004b] Microsoft Corporation, http://www.microsoft.com, Retrieved: 2004

[MS2004c] Microsoft Corporation, Microsoft Virtual Server 2005 Technical Overview White Paper, http://www.microsoft.com/windowsserversystem/virtualserver/overview/vs2005tech.mspx, 2004

[MS2004d] Microsoft Corporation, Virtual Server 2005 Administration Guide, http://www.microsoft.com/technet/prodtechnol/virtualserver/2005/proddocs/default.mspx, Retrieved: 2004

[MS2004e] Microsoft Corporation, Stop Server Sprawl - Server Consolidation Can Increase Your Business Agility and Operational Efficiency, http://www.microsoft.com/business/executivecircle/content/printPage.aspx?cId=887&subcatID=4, Retrieved: 2004

[MS2005a] Microsoft Corporation, Microsoft Windows, www.microsoft.com/windows/default.asp, Retrieved: 2005

[MS2005b] Microsoft Corporation, Microsoft SQL Server, www.microsoft.com/sql/default.asp, Retrieved: 2005

[MSQ2005] MySQL AB., MySQL 5.0 Database Server, http://www.mysql.com/products/database/, Retrieved: 2005

[MST+2005] Menon, A., Santos, J. R., Turner, Y., Janakiraman, G., Zwaenepoel, W., Diagnosing Performance Overheads in the Xen Virtual Machine Environment, Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments pages 13-23, 2005

[OC2005a] Oracle Corporation, Oracle Database, http://www.oracle.com/database/index.html, Retrieved: 2005

[OC2005b] Oracle Corporation, Oracle Application Server, http://www.oracle.com/appserver/index.html, Retrieved: 2005

[OSD2005] The Open Source Database Benchmark, http://osdb.sourceforge.net/, Retrieved: 2005

[PG1974] Popek, G. J. and Goldberg, R. P., Formal Requirements for Virtualizable Third Generation Architectures, Communications of the ACM Volume 17 number 7, 1974

[PH2002] Pepple, K., Hornby, D., Consolidation in the Data Center: Simplifying IT Environments to Reduce Total Cost of Ownership, Prentice Hall, 2002

[PM2002] Mehra, P., Global Deployment of Data Centers, IEEE Internet Computing, September/October 2002, http://ieeexplore.ieee.org/iel5/4236/22232/01036036.pdf, Retrieved: 2004

[PP1973] Parnas, D. L., and Price, W. R., The Design of the Virtual Memory Aspects of a Virtual Machine, Proceedings of the Workshop on Virtual Computer Systems, pages 184-190, 1973

[PT2004] Price, D., and Tucker, A., Solaris Zones: Operating System Support for Consolidating Commercial Workloads, LISA XVIII: Large Installation System Administration Conference, 2004

[RG2005] Rosenblum, M., and Garfinkel, T., Virtual Machine Monitors: Current Technology and Future Trends, IEEE Computer May 2005, pages 39-47

[RI2000] Robin, J. S., Irvine, C. E., Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor, Proceedings of the 9th USENIX Security Symposium, pages 129-144, 2000

[RMS+2000] Reumann, J., Mehra, A., Shin, K.G., and Kandlur, D., Virtual Services: A New Abstraction for Server Consolidation, Proceedings of 2000 USENIX Annual Technical Conference, 2000

[Ros2004] Rosenblum, M., The Reincarnation of Virtual Machines, ACM Queue, Volume 2 number 5, pages 34-40, 2004

[SB2005] SysBench: A System Performance Benchmark, http://sysbench.sourceforge.net/, 2005

[Sch2003] Schneider, G.P., Electronic Commerce, Thomson Publishers, 2003

[SHH+2005] Surányi, P., Hirotake, A., Hirotsu, T., Yasushi, S., Kazuhiko, K., General Virtual Hosting via Lightweight User-level Virtualization, Proceedings of the 2005 Symposium on Applications and the Internet, pages 229-236, 2005

[SI2005] SPARC International, http://www.sparc.org/, Retrieved: 2005

[Sim2004] The Siemon Company, Siemon 10G ip Data Center Solution, http://www.siemon.com/us/white_papers/03-10-10-data_centers.asp, Retrieved: 2004

[SM2000] Sun Microsystems, Inc., Scaling the N-Tier Architecture, http://wwws.sun.com/software/white papers/wp-ntier/wp-ntier.pdf, 2000, Retrieved: 2004

[SM2001] Sun Microsystems, Inc., Sun Fire 6800, 4810, 4800, and 3800 Systems Dynamic Reconfiguration User Guide, http://docs.sun.com/db/doc/806-6783-10, 2001, Retrieved: 2004

[SM2003a] Sun Microsystems, Inc., Sun Management Center 3.5 Version 2 Supplement for Sun Fire 15K/12K Systems, Revision A, http://docs-pdf.sun.com/817-3624/817-3624.pdf, 2003, Retrieved: 2004

[SM2003b] Sun Microsystems, Inc., Sun Fire 15K/12K Systems Overview, ftp://docs-pdf.sun.com/806-3509-12/806-3509-12.pdf, 2003, Retrieved: 2004

[SM2004a] Sun Microsystems, Inc., Reality Check, http://www.sun.com/executives/realitycheck/reality-051404.html, Retrieved: 2004

[SM2004b] Sun Microsystems, Inc. , N1 Grid: Managing n Computers as 1, http://wwws.sun.com/software/solutions/n1/overview.html, Retrieved: 2004

[SM2004c] Sun Microsystems, Inc. , Vision, http://wwws.sun.com/software/solutions/n1/vision.html, Retrieved: 2004

[SM2004d] Sun Microsystems, Inc., http://www.sun.com, Retrieved: 2004

[SM2004e] Sun Microsystems, Inc., Solaris Containers (formerly N1 Grid Containers) - Optimizing Resource Utilization to Deliver Predictable Service Levels, http://wwws.sun.com/software/solaris/10/ds/containers.jsp, Retrieved: 2004

[SM2004f] Sun Microsystems, Inc., Dynamic Reconfiguration for Sun Fire Midrange Servers, http://www.sun.com/servers/midrange/dr_sunfire/, Retrieved: 2004

[SM2004g] Sun Microsystems Inc., N1 Grid: n Computers Operating as 1. Essays - Jonathan Schwartz: How to Make IT Infrastructure More Responsive to Business Needs, http://www.sun.com/software/solutions/n1/essays/schwartz.html, Retrieved: 2004

[SM2005a] Sun Microsystems, Inc., Installing and Administering Solaris Container Manager 1.1, ftp://docs-pdf.sun.com/817-7551/817-7551.pdf, Retrieved: 2005

[SM2005b] Sun Microsystems Inc., Solaris 10, http://www.sun.com/software/solaris/, Retrieved: 2005

[SM2005c] Sun Microsystems Inc. Sun Fire E25K Specifications, http://www.sun.com/servers/highend/sunfire_e25k/specifications.jsp, Retrieved: 2005

[SM2005d] Sun Microsystems, Inc., System Administration Guide: Solaris Containers—Resource Management and Solaris Zones http://docs-pdf.sun.com/817-1592/817-1592.pdf, Retrieved: 2005

[SM2005e] Sun Microsystems, Inc., LibMicro: Portable Microbenchmarks, http://www.opensolaris.org/os/community/performance/libmicro/, Retrieved: 2005

[SM2005f] Sun Microsystems, Inc., Sun Fire V40z Server, http://www.sun.com/servers/entry/V40z/index.jsp, Retrieved: 2005

[SM2005g] Sun Microsystems, Inc., Sun Fire V240 Server, http://www.sun.com/servers/entry/v240/, Retrieved: 2005

[SM2005h] Sun Microsystems, Inc., OpenSolaris Community: ZFS, http://www.opensolaris.org/os/community/zfs/, Retrieved: 2005

[SN2005] Smith, J. E., and Nair, R., Virtual Machines: Versatile Platforms for Systems and Processes, Morgan Kaufmann Publishers, San Francisco, 2005

[SS2000] Sullivan, D. G., Seltzer, M. I., Isolation with Flexibility: A Resource Management Framework for Central Servers, Proceedings of the 2000 USENIX Annual Technical Conference, pages 337-350, 2000

[STY+2002] Shen, K, Tang, H., Yang, T., Chu, L., Integrated Resource Management for Cluster-based Internet Services, 5th Symposium on Operating Systems Design and Implementation, 2002

[SV2002] Sliwa, C., and Vijayan, J., (Computerworld) IT Managers Tackle Windows Server Sprawl, http://www.computerworld.com/softwaretopics/os/story/0,10801,75272,00.html, 2002, Retrieved: 2004

[SVL2001] Sugerman, J., Venkitachalam, G, and Lim, G., Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, Proceedings of the 2001 USENIX Annual Technical Conference, pages 1-14, 2001

[TC2004] Tucker, A., Comay, D., Solaris Zones: Operating System Support for Server Consolidation, 3rd Virtual Machine Research and Technology Symposium, 2004

[TOG2005] The Open Group, http://www.opengroup.org/certification/unix-home.html, Retrieved: 2005

[UM2005] UMLinux, http://www.umlinux.de, Retrieved: 2005

[UML2005] User Mode Linux, http://user-mode-linux.sourceforge.net, Retrieved: 2005

[UNR+2005] Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F.C.M., Anderson, A. V., Bennett, S. M., Kägi, A., Leung, F. H., Smith, L., Intel Virtualization Technology, IEEE Computer, pages 48-56, May 2005

[UVS+2004] Uhlig, V., LeVasseur, J., Skoglund, E., and Dannowski, U., Towards Scalable Multiprocessor Virtual Machines, Proceedings of the 3rd Virtual Machine Research & Technology Symposium (VM'04), 2004

[VGR1998] Verghese, B., Gupta, A., and Rosenblum, M., Performance Isolation: Sharing and Isolation in Shared-Memory Multiprocessors, Proceedings of 8th International Conference on Architectural Support for Programming Languages and Operating Systems, pages 181-192, 1998

[Vir2005] Virtuozzo, http://www.virtuozzo.com, Retrieved: 2005

[VMW2004a] VMWare, Inc., http://www.vmware.com, Retrieved: 2004

[VMW2004b] VMWare, Inc., Virtual Infrastructure, http://www.vmware.com/vinfrastructure/, Retrieved: 2004

[VMW2004c] VMWare, Inc., VMware VirtualCenter 1.1, http://www.vmware.com/products/vmanage/vc_features.html, Retrieved: 2004

[VMW2004d] VMware, Inc., VMware Virtual SMP, http://www.vmware.com/products/server/vsmp_features.html, Retrieved: 2004

[VMW2004e] VMware Inc., VMware ESX Server FAQ, http://www.VMware.com/products/server/esx_faqs.html, Retrieved: 2004

[VMW2005a] VMware Inc., VMware ESX Server Administration Guide, 2005

[VMW2005b] VMware Inc., Systems Compatibility Guide for ESX Server 2.x, http://www.vmware.com/pdf/esx_systems_guide.pdf, Retrieved: 2005

[VMW2005c] VMware Inc., VMware ESX Server Guest Operating System Installation Guide, http://www.vmware.com/pdf/GuestOS_guide.pdf, Retrieved: 2005

[VMW2005d] VMware Inc., VMware VMotion
http://www.vmware.com/products/vc/vmotion.html, Retrieved: 2005

[VMW2005e] VMware Inc., VMware GSX Server,
http://www.vmware.com/products/gsx/, Retrieved: 2005

[Wal1995] Waldspurger, C. A., Lottery and Stride Scheduling: Flexible Proportional Shares Resource Management, Ph.D. Thesis, Massachusetts Institute of Technology, 1995

[Wal2002] Waldspurger, C. A., Memory Management in VMware ESX Server, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, pages 181-194, 2002

[WCS+2004] Whitaker, A., Cox, R. S., Shaw, M., and Gribble, S. D., Constructing Services with Interposable Virtual Hardware, Proceedings of the First Symposium on Networked Systems Design and Implementation, pages 169-182, 2004

[Wil2005a] Williamson, M., Extreme Paravirtualisation: Beyond Arch/Xen, http://www.cambridge.intel-research.net/~mwilli2/proposal_final.pdf, Retrieved: 2005

[WSG2002a] Whitaker, A., Shaw, M. and Gribble, S. D., Scale and Performance in the Denali Isolation Kernel, SIGOPS Operating System Review, pages 195-209, 2002

[WSG2002b] Whitaker, A., Shaw, M., and Gribble, S. D., Dinali: Lightweight Virtual Machines for Distributed and Networked Applications, University of Washington Technical Report, 2002