

# **Anticipatory Models of Load Balancing in Cloud Computing**

A Thesis Submitted for the Degree of  
Doctor of Philosophy  
By  
Shahrzad Aslanzadeh



University of Technology Sydney  
New South Wales, Australia

## CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student: *Shahrzad Aslanzadeh*

Date: 04/14/2016

## **Acknowledgment**

I would like to express my sincere gratitude to my supervisor Dr. Zenon Chaczko for his support, continuous guidance, meticulous suggestions and his inexhaustible patience especially during preparation of this dissertation. At many stages in the course of this research, I benefited from his advice, particularly so when exploring new ideas.

Also I would like to thank Dr. Christopher Chiu, who as a good friend was always willing to help and give his best suggestions. Many thanks to Christopher Mcdermid for helping me through my case study design.

I will be thankful to my close friends for their friendly support and caring.

I express my personal love and appreciation to my mum, dad and my sister for constant motivation and support.

Finally, I would like to thank my husband Alireza, who was always there cheering me up and stood by me through the good times and the bad.

## **Abstract**

Cloud Computing is a recent arrival to the world of IT infrastructure. The concept allows companies to maximise utilisation of their potentials and consequently boost their performance. One of the main benefits of Cloud Computing is the significant increase in efficiency of executing business plans. Additionally, Cloud Computing provides large-scale applications with powerful computing power across global locations. Yet Cloud users are able to share their data easily by using replication methodologies.

Cloud Computing structure has been developed based on a multi-tenancy concept. Therefore, availability and efficiency of the resources are important factors in the Cloud architecture. However, as the numbers of users are increasing rapidly, the load will have a significant impact on performance and operation of the Cloud systems. Accordingly, optimised load balancing algorithms that can manage the Cloud load in a time- and cost-efficient manner are required.

Much research in recent years has been dedicated to optimising load balancing in Cloud Computing. This optimisation is demonstrated through a balanced network of interacting resources. The goal of this network is to minimise the wait time and maximise utilisation of the throughput.

This thesis provides a set of solutions which mitigate the problem of load balancing in the Cloud. The dissertation investigates a novel class of heuristic scheduling algorithms that improves load balancing in workflow scheduling applications.

Furthermore, it proposes a new anticipatory replication methodology with the objective of improving data availability to enhance the load balancing between the Cloud sites.

In summary, this research innovation implicates the design of optimised load balancing algorithms that consider the magnitude and direction of the load in workflow applications. Furthermore, by architecting the anticipatory replication algorithm, it minimises the numbers of the replicas and enhances the effective network usage in Cloud-based systems.

## Contents

I. Principal Theory and Concepts .....	2
Background .....	3
1.1 Introduction .....	3
1.2 Rational .....	4
1.3 Research Contribution.....	6
1.4 Research Questions .....	6
1.5 Formulation of the Hypothesis .....	8
1.5.1 Approach and Hypothesis Validation.....	9
1.5.1.1 Research Action Study .....	9
1.5.1.2 Experiments.....	9
1.5.2 Expected Outcomes.....	10
1.6 Outline of the Thesis .....	11
1.7 Related Publications.....	17
Literature Review.....	19
2.1 Cloud Computing .....	19
2.1.1 Cloud Characteristics .....	24
2.1.2 Cloud Taxonomy .....	25
2.1.3 Cloud Architecture .....	26
2.1.4 Cloud Stack .....	28
2.1.4.1 Infrastructure as a Service (IaaS) .....	28
2.1.4.1.1 Characteristics of IaaS.....	29
2.1.4.2 Platform as a Service (PaaS) .....	29
2.1.4.2.1 Characteristics of PaaS.....	30
2.1.4.3 Software as a Service (SaaS).....	30
2.1.4.3.1 Characteristics of SaaS.....	30
2.2 Cloud Computing: Challenges and Benefits .....	31
2.3 Load Balancing in Cloud Computing.....	34
2.3.1 Load Balancing: Concept, Definition, Benefits, Challenges.....	34
2.3.2 Load Balancing Measurement Criteria.....	35

2.3.3 Load Balancing Methods.....	36
2.4 Focus of This Research .....	38
2.4.1 Load Balancing Algorithms: Independent vs Dependent.....	38
2.4.1.1 Dependent Load Balancing Algorithms .....	39
2.4.1.2 Independent Load Balancing Algorithms.....	45
2.5 The Quest of This Research .....	49
2.5.1 The Anticipatory Approach.....	49
2.5.2 Workflow Scheduling.....	51
2.5.3 Replication Strategy .....	53
2.5.4 Research Issues .....	55
2.5.4.1 Workflow Scheduling.....	55
2.5.4.2 Replication Methodology .....	55
2.6 Summary .....	58
Research Methodology.....	59
3.1 Research Design.....	59
3.2 Research Method.....	60
3.2.1 Research Rules .....	62
3.3 Modelling Approaches .....	62
3.3.1 Black-box Modelling Approaches.....	62
3.3.2 White-Box Modelling Approaches .....	63
3.3.3 Gray-Box Modelling Approach.....	63
3.4 Experimental Methodologies Overview.....	65
3.4.1 Anticipatory Approach.....	65
3.4.2 Spring Tensor Model.....	65
3.4.2.1 Mathematical Apparatus.....	66
3.4.3 Anticipatory Replication Methodology .....	69
3.5 Summary of the Chapter .....	70
Research Action Study .....	71
4.1 Introduction .....	71
4.2 Load Balancing .....	72
4.3 Presence.....	75

4.4	Cloud Metrics.....	77
4.5	The Case Study: HDM Load Monitoring.....	79
4.5.1	Load Monitoring Tool.....	81
4.5.2	HDM Load Tolerance .....	83
4.5.2.1	Testing Tool .....	83
4.5.2.2	Page Analyser.....	84
4.5.2.3	Load Test-10 Virtual Users .....	86
4.5.2.3.1	10 Users-Load time vs Clients Active.....	86
4.5.2.3.2	10 Users - Number of the Active Requests vs Clients Active.....	86
4.5.2.3.3	10 Users - HDM Content Type Distribution .....	87
4.5.2.3.4	10 Users HDM Content Load Distribution .....	87
4.5.2.4	Load Test - 20 Virtual Users.....	88
4.5.2.4.1	20 Users-Load Time Vs Clients Active.....	88
4.5.2.4.2	20 Users- Number of the Active Requests Vs Clients Active.....	88
4.5.2.4.3	20 Users - HDM Content Type Distribution .....	89
4.5.2.4.4	20 Users HDM Content Load Distribution .....	89
4.5.2.5	Load Test - 50 Virtual Users.....	90
4.5.2.5.1	50 Users-Load Time Vs Clients Active.....	90
4.5.2.5.2	50 Users -Number of the Active Requests vs Clients Active.....	90
4.5.2.5.3	50 Users- HDM Content Type Distribution .....	91
4.5.2.5.4	50 Users HDM Content Load Distribution .....	91
4.5.2.6	Load Test - 100 Virtual Users.....	92
4.5.2.6.1	100 Users-Load Time Vs Clients Active.....	92
4.5.2.6.2	100 Users- Number of the Active Requests Vs Clients Active.....	92
4.5.2.6.3	100 Users - HDM Content Type Distribution .....	93
4.5.2.6.4	100 Users HDM Content Load Distribution .....	93
4.5.2.7	Results Analysis .....	94
4.5.2.7.1	Average Load Time Results .....	94
4.5.2.7.2	HDM Content Type Distribution Results.....	94
4.5.2.7.3	HDM Content Load Distribution Results.....	96
4.6	Overall Analysis and Proposed Solution.....	97
4.7	Conclusion.....	98
II.	Contribution to Research.....	99

STEM-PSO Based Task Scheduling Algorithm .....	100
5.1 Introduction .....	101
5.2 Load Balancing Problem Formulation .....	102
5.3 STEM-PSO Scheduling Algorithm.....	104
5.3.1 Particle Swarm Optimisation.....	104
5.3.2 STEM Algorithm.....	105
5.3.3 The Fundamentals of STEM-PSO.....	111
5.4 Experiment Results .....	112
5.4.1 Total Execution Time.....	113
5.4.2 CPU Load and CPU Time.....	114
5.4.3 Memory Rate.....	115
5.4.3.1 Workflow Soft Error Rates.....	116
5.5.4 Comparison of STEM-PSO Results.....	117
5.5.5 Experiment Analysis .....	118
5.5 Conclusion.....	119
Load balancing & Data Replication Strategy.....	120
6.1 Introduction .....	120
6.2 Problem formulation .....	122
6.3 Proposed Methods.....	123
6.3.1 SDDRC Architecture Design .....	123
6.3.2 SDDRC Algorithm.....	125
6.4 Results and Analysis .....	131
6.4.1 Mean Job Execution Time.....	133
6.4.2 Effective Network Usage .....	134
6.4.3 Total Number of the Replicas .....	135
6.4.3.1 Applying Soft Error Rates on Memory .....	136
6.4.4. Analysis of SDDRC Results.....	137
6.4.5 Experiment Analysis .....	138
6.5 Conclusion.....	139
Conclusion.....	140
7.1 Review.....	140



7.2 Thesis contribution.....	141
7.2.1 Discussion .....	141
7.2.2 Limitations .....	143
7.3 Future work .....	144
7.4 Final Remarks .....	146
III. Bibliography and publications.....	147
Bibliography.....	148
Appendix .....	175
9.1 Experiment Specifications.....	175
9.1.1 HDM System.....	175
9.1.1.1 Development specifications .....	176
9.1.2 STEM-PSO Implementation Details .....	179
9.1.3 Cloudsim Specification Details.....	183
9.1.3.1 Adding a new file in Tier structure.....	188

## List of Figures

Figure 1.1- Thesis outline.....	14
Figure 2.1-Litriture review high-level mind map.....	20
Figure 2.2- IT evolution (KPMG 2011).....	21
Figure 2.3- Comparing Cloud Computing, Cluster Computing and Grid Computing (Modified from Google trend).....	22
Figure 2.4 -Cloud taxonomy .....	26
Figure 2.5 -Reference model of Cloud architecture, adopted from, (Buya et al. 2008).....	27
Figure 2.6- Cloud stack categorisation, adopted from (Barkat et al. 2014) .....	28
Figure 2.7 -Load balancing categories .....	36
Figure 3.1 - Cloudsim architecture (Adopted from Calheiros 2011) .....	61
Figure 3.2 - Thesis modelling approaches (Adopted from DTU compute website) .....	64
Figure 3.3- Elastic sub-network model .....	67
Figure 4.1-Server availability model (Chaczko et al. 2014).....	73
Figure 4.2- Hybrid load balancing model (Chaczko et al. 2014).....	74
Figure 4.3- HDM System .....	79
Figure 4.4- Conceptual Architecture, Data Centric Model (Chaczko et al. 2011).....	80
Figure 4.5- High-level HDM database design.....	81
Figure 4.6- HDM load balancing model (Chaczko et al. 2011) .....	82
Figure 4.7- System health monitoring UI.....	83
Figure 4.8- Load time vs clients active - 10 virtual users.....	86
Figure 4.9- Requests per second vs clients active – 10 virtual users.....	86
Figure 4.10- HDM content type distribution - 10 virtual users .....	87
Figure 4.11-HDM content load distribution - 10 virtual users .....	87
Figure 4.12- Load time vs clients active - 20 virtual users.....	88
Figure 4.13- Requests per second vs clients active – 20 virtual users.....	88
Figure 4.14- HDM content type distribution - 20 virtual users .....	89
Figure 4.15- HDM content load distribution -20 virtual users .....	89
Figure 4.16- Load time vs clients active - 50 virtual users.....	90
Figure 4.17- Requests per second vs clients active – 50 virtual users.....	90
Figure 4.18-HDM content type distribution - 50 virtual users .....	91
Figure 4.19- HDM content load distribution -50 virtual users .....	91
Figure 4.20-Load time vs clients active - 100 virtual users.....	92
Figure 4.21- Requests per second vs clients active – 100 virtual users.....	92
Figure 4.22- HDM content type distribution - 100 virtual users .....	93
Figure 4.23 - HDM content load distribution - 100 virtual users .....	93
Figure 5.1- Example of workflow modelling.....	103
Figure 5.2- STEM workflow parameters (Bond, angle, dihedral, non-local interaction) .....	107
Figure 5.3- Sample tasks allocations on (PC1-PC4) .....	112
Figure 5.4- Comparison of total execution time between STEM-PSO and HEFT .....	113
Figure 5.5- Comparison of total CPU Time between STeM-PSO and HEFT .....	114
Figure 5.6-Comparison of total Memory utilisation rate between STEM-PSO and HEFT .....	115
Figure 6.1– High level SDDRC system architecture.....	124

Figure 6.2- GRMS high-level architecture.....	125
Figure 6.3- GRMS Tier insertion .....	126
Figure 6.4-Creating the VM data inputs in Cloudsim .....	132
Figure 6.5 -Initiaing the VM creating at runtime .....	132
Figure 6.6- Mean job execution time .....	133
Figure 6.7- Effective network usage .....	134
Figure 6.8 -Total number of replications.....	135
Figure 9.1-HDM system high-level design .....	175
Figure 9.2- STEM-PSO high-level logical design .....	179
Figure 9.3-Pegasus workflow data generator .....	180
Figure 9.4- STEM-PSO front end .....	180
Figure 9.5- Hessian Evaluation Rate .....	181
Figure 9.6- PSO algorithm input -front end .....	181
Figure 9.7- STEM-PSO main class code.....	182
Figure 9.8- Cloudsim high-level design .....	183
Figure 9.9- Cloudsim class architectures .....	185
Figure 9.10-Cloudsim submission time frontend .....	187
Figure 9.11- Inserting a node in Tier structure.....	188
Figure 9.12- Updating the pointer place in Tier structure .....	189

## List of Tables

Table 1.1- Thesis part I and part II structure .....	12
Table 2.1- Benefits of the Cloud Computing .....	32
Table 2.2 -Challenges in the Cloud Computing .....	33
Table 2.3- Summary of the reviewed dependent scheduling algorithms .....	44
Table 2.4- Summary of the reviewed independent algorithm .....	48
Table 2.5- Summary of the reviewed workflow scheduling algorithms .....	56
Table 2.6- Summary of the reviewed replications methodologies .....	57
Table 4.1- VM properties .....	85
Table 4.2- Summary of No. of clients vs average load time.....	94
Table 4.3-HDM content type distribution results – 10, 20,50, 100 virtual users.....	95
Table 4.4-HDM average content type load time distribution results 10,20,50,100 virtual users	96
Table 4.5- Average user load time vs average html load time .....	97
Table 5.1- Interpretation of bond, angle, dihedral and non-local connections between tasks on workflow model (Aslanzadeh & Chaczko 2015). .....	108
Table 5.2 -CPU load utilisation rate using STeM-PSO.....	114
Table 5.3-Analysing the impact of soft error rates on Memory rate .....	116
Table 5.4- Total workflow execution time applications using STEM-PSO and HEFT .....	117
Table 5.5- CPU utilisation rate of workflow applications using STEM-PSO and HEFT .....	117
Table 5.6- Memory utilisation rate of workflow applications using STEM-PSO and HEFT ..	117
Table 6.1 -Analysing the impact of soft error rates on Memory rate .....	136
Table 6.2 - Mean job execution time using SDDRC (in milliseconds).....	137
Table 6.3 - Effective network usage using SDDRC (in percentage).....	137
Table 6.4 - Total number of replication using SDDRC.....	137

## **Glossary**

AMAZON EC2	Amazon Elastic Cloud Computing
ANM	Anisotropic Network Model
DAG	Directed Acyclic Graph
DPSO	Discrete Version Of PSO
ENM	Elastic Network Models
FCFS	First Come First Served
FMOC	Finite Multi-Order Context
GA	Genetic Algorithm
GBEST	Global Best Position
GNM	Gaussian Network Model
GRMS	Global Replica Management System
HDM	Hospital Data Management
HEFT	Heterogeneous Earliest Finish Time
HTML	Hyper Text Mark-up Language
I/O	Input/Output
IAAS	Infrastructure As A Service
ICT	Information & Communication Technology
IOT	Internet Of Things
IT	Information Technology
LFU	Least Frequently Used
LRU	Least Recently Used
MAKESPAN	Total Length Of Schedule
MBIT/S	Megabit Per Second
NIST	National Institute Of Standards And Technology
NP PROBLEM	Non-Deterministic Polynomial Time
PAAS	Platform As A Service
PBEST	Best Local Position
PSO	Particle Swarm Optimisation
QOS	Quality Of Service
RC	Relative Cost

SAAS	Software As A Service
SDDRC	Smart Dynamic Data Replication In Cloud Computing
SHEFT	Scalable Heterogeneous Earliest Finish Time
SLA	Service Level Agreement
STEM	Generalised Spring Tensor Model
STEM-PSO	Generalised Spring Tensor Model- Particle Swarm Optimisation
URL	Uniform Resource Locator
VM	Virtual Machine
XML	Extensible Mark-up Language
XMPP	Extensible Messaging And Presence Protocol

# **I. Principal Theory and Concepts**

# Chapter 1

## Background

*“Everything is going to be connected to Cloud and data. All of this will be mediated by software”- Satya Nadella-1967*

Chapter 1 illustrates the high-level overview of the thesis framework. It explains the concepts and rationale of the dissertation followed by research contributions and research questions. The chapter also formulates the hypothesis of the experimental work. Finally, it concludes by providing an outline overview of the existing chapters.

### 1.1 Introduction

Over the past decade, industries and academia have increased their expectations for more robust computing techniques. In the field of IT, improvements in the Internet and also developments in economical IT infrastructure have been the main key in reshaping the users' requirements (Metcalfe 2000). Thus, creating more-powerful computational models has become a necessity. Cloud Computing is an example of newly created structures for improving Internet services such as hosting and delivering (Zhang, Cheng & Boutaba 2010).

Cloud Computing is generally based on virtualisation techniques with the capability of providing automated services (Ramezani et al. 2013). These include resource management and load balancing, which are developed and/or executed according to the user's requirements, regardless of location and time zone (Grant et al. 2013).

Cloud Computing can address the business infrastructure requirements, such as scalability and elasticity, when future changes occur (Armbrust et al. 2009). Cloud



Computing provides a valuable tool for businesses, as with its use, organisations do not need to be concerned about wasting their resources by under- or over-estimating the future needs. In other words, the required resources can be allocated or released as future demand unfolds. Businesses can reduce the costs associated with their infrastructure by employing the pay-as-you-go model, which is utilised for charging the Cloud users (Gupta et al. 2013),(Rimal et al. 2009) ,(Yike, Ghanem & Rui 2012).

In recent years, the number of Cloud users has risen sharply and as such, technical aspects of Cloud Computing have received increased attention. Cloud load measuring methods are the primary examples of such technical issues, which pose new challenges for Cloud service providers (Rahman et al. 2014).

In order to overcome some of the difficulties associated with load analysis, researchers and development agencies have proposed a variety of techniques, algorithms, and architectures (Wang et al. 2013). Each Cloud-based system, however, includes a complex architecture and requires customised specifications. This fact, in turn, creates complications in developing a standard design for load balancing management. Therefore, load balancing optimisation approaches play a crucial role in the world of Cloud infrastructures. Such load balancing techniques would be capable of distributing the load evenly among all available resources (Sahu et al. 2013).

## **1.2 Rational**

Business enterprises and academia have proposed various load balancing algorithms as evidence in the scientific literature. Such computational solutions are valuable tools in dealing with load management (Ardanga et al. 2011). However, they do not address how application structures and associated interrelationship can be modelled when dealing with the behaviour of the load balancing design.

In recent years, many research studies have been focused on the load balancing algorithms in Cloud Computing. Prime examples include static scheduling, dynamic and hybrid algorithms (Ghutke & Shrawankar 2014).

There have been, however, few studies on load balancing algorithms that focus on their dependent patterns, only. The reason behind the lack of studies in this area could be associated with complex nature of the workflow architecture; as well as, their relatively unpredictable behaviour when it comes to the resource management (Barrett, Howley & Duggan 2011).

Workflow scheduling is a valuable tool for applications used in e-business and e-science (Yilin et al. 2012). Processing of data in such applications requires a complex procedure, thus emphasising the importance of workflow scheduling algorithms.

Cloud Computing platforms can facilitate utilisation of e-science and e-business applications. Considering the data intensive nature of these applications, such as weather forecasting and online booking, complex data processing is required. Additionally, to improve reliability and performance of the e-science applications and in line with the Cloud system architecture designs, Li & Mascagni (2013) underlined development of an effective workflow scheduling algorithm.

Considering workflow applications, completion of one task is required for execution of interconnected tasks and subsequently for the execution of interconnected jobs (Singh & Singh 2013). Therefore, developing smart algorithms which are capable of recognising the behaviour of the interconnected tasks and jobs is vital. Such smart computational models can then address the system constraints and user requirements, enabling better performance (Chun-Chen et al. 2008).

Many recent research studies have been focused on load balancing techniques and their application to Cloud Computing. One of these techniques is optimising data retrieval, which has a major impact on load balancing (Shaw et al. 2014). The data replication technique, however, as an important data retrieval method, has not been studied deeply in the field of Cloud Computing. Complexity in embedding replication methodology in Cloud-based systems may be the reason for the lack of attention to this subject (Anikode & Bin 2011). Such a lack of attention emphasises the need for developing a smart dynamic replication approach, capable of optimising load balancing through replication.

This research sheds light on workflow scheduling and the data replication mechanisms in Cloud Computing.

## **1.3 Research Contribution**

The value of this research should be perceived as:

- Architecting a robust load balancing algorithm with anticipatory behaviour which is platform independent and is designed for Cloud Computing infrastructure.
- Investigating the research action studies that focus on load balancing in Cloud Computing.
- Understanding the behaviour of the data intensive applications by analysing and modelling the inter-connectivity of their tasks.
- Identifying the impact of anticipatory replication on load balancing by analysing the replication strategies in Cloud-based systems.

## **1.4 Research Questions**

The overall aim of this research is to evaluate the following concepts:

- Apply the magnitude and direction of the load between interconnected tasks for optimising the load balancing behaviour in Cloud Computing.
- Pre-replicate the files with high access probability in local servers before provisioning requests have been submitted.

Identifying the loads' fluctuations will lead to forecasting the future load behaviour between interconnected tasks. Moreover, analysing load variations will be useful in terms of identifying the anomalies and threats in workflow applications that could lead to effective decision making.

To satisfy the workflow load balancing behaviour, a heuristic workflow scheduling algorithm will be investigated to identify the load fluctuations among Cloud networks.

A review of the literature, documented in chapter 2 of this thesis, illustrates the heuristic computational models that have been proposed to improve the load balancing approaches in Cloud Computing by analysing the static and dynamic movement of the tasks (Madivi & Kamath 2014). Yet a shortage remains of effective tools to capture the load fluctuations of the workflow tasks that can project the interactions and dependencies between interconnected tasks.

Anticipatory replication will optimise the load balancing by pre-replicating the files that have high access probability in the future. The pre-replication will be useful in terms of minimising the total job execution time and effective network usage rate. In this method, the anticipatory function of the replication method can provide the analytical base for load balancing optimisation.

Specific research questions addressed in this investigation are:

1. Given the importance of the scientific applications, what makes the Cloud system adaptive to dynamic conditions related to workflow scheduling?
2. Given the importance of data retrieval, how does the anticipatory data replication model impact load balancing in Cloud Computing?
3. Given the need for workflow scheduling and data replication, how can these methods impact the anticipatory behaviour of the Cloud-based systems?
4. Given that some of the important load balancing algorithms have been derived from mathematics, are limitations from theory consistent with simulation results?

The expected benefits of this research are credited on the fact that the behaviour of the workflow load balancing approaches can be regulated based on the interactions between tasks and jobs within their local neighbourhood and non-neighbourhood situations. Moreover, it is anticipated that the proposed anticipatory replication design will make a contribution to situate the research theme in the design of an effective load

balancing algorithm. The results provide more enhanced resource usage and data processing technologies suitable for computing complex applications.

## 1.5 Formulation of the Hypothesis

Hypothesis: How can a heuristic anticipatory approach combined with data replication strategies improve the load balancing in Cloud to ensure its availability?

According to the hypothesis, two main essential points will be considered in details within this doctoral dissertation:

1. Heuristic methodology:

The solution to the load balancing issue within workflow scheduling can be found in biological systems coordinating the dependencies of the interconnected tasks. By applying the meta-heuristic methods, it is possible to calculate the magnitude and direction of the load between interconnected tasks and integrate that in load balancing optimisation methodology.

To achieve a global perspective of load balancing in Cloud-based systems, the anticipatory behaviour of interconnected tasks will be investigated. The matter shall be examined as a gray-box model to determine the impact of the load changes throughout the workflow application network.

2. Replication strategy:

The aim of this approach is to anticipate replicating the high probable files that may be accessed later. The provisioning of the pre-replications provides end-users with a minimum access time that ultimately minimises the total task execution time.

In the same fashion, for effective network usage, the anticipatory-replication strategy will minimise the number of the replicated files, and as a result the number of files to be transferred to the targeted servers will be reduced. The anticipatory behaviour of pre-replication techniques is achievable through gray-

box modelling by applying the mathematical threshold embedded within the replication design.

## **1.5.1 Approach and Hypothesis Validation**

### **1.5.1.1 Research Action Study**

The action study investigates the performance of the hospital data management (HDM) Cloud-based system by identifying the existing load balancing challenges within its framework.

Online load balancing tools have been incorporated to assess the current HDM load balancing functionality. The results then have been used for development of novel load monitoring services.

The developed tool reflects the results of applying load balancing techniques using presence protocols to indicate the health status of the systems' elements in terms of load changes.

### **1.5.1.2 Experiments**

The aim of the experimental work is to design and implement a series of practical solutions to find an optimised load balancing method suitable for Cloud-based systems.

We have emulated the Cloud system with a Cloud simulation tool and Java programming which is built upon the heuristic and replication functions, necessary to implement the experiments.

As the heuristic functions are executed, the real-time load optimisation is pictured in a real simulation environment.

The designed work is suitable for predictive analytics. Users are able to adjust the parameters based on the experimental environment. The front-end shows how the results may be affected by occurring changes in heuristic parameters. The experimental outcome validates the anticipatory behaviour of the system, which measures the impact of changes on magnitude and direction of the load.

Applying the mathematical apparatus of the replication strategy, the load balancer will anticipate the files that will be needed by users in future. The outcome of the experiment will validate the anticipatory behaviour of the system by pre-replicating the data with high access probability.

In summary, to find the answers for designated research questions, experiments are completed through gray-box design to present the optimised solution for load balancing in Cloud Computing. The details of the algorithms applied in the experiments are depicted in the research methodology chapter.

## **1.5.2 Expected Outcomes**

The results confirm that the heuristic approaches in both experiments represent an improved method of load balancing in Cloud-based systems. In particular, encapsulating the heuristic designs in a workflow scheduling model in Cloud Computing provides an optimised load balancing method that results in minimised makespan along with optimised memory and CPU rates. Additionally, the heuristic anticipatory-replication methodology completes the submitted jobs with minimum completion time while it minimises the replication numbers and improves the effective network usage.

In summary, the expected outcomes validate the anticipatory function embedded in both methodologies. The results anticipate the load fluctuations in workflow structures and forecast the potential high accessible files needed for replications.

## 1.6 Outline of the Thesis

As table 1.1 on page 12 shows, the thesis is divided into two main parts;

- Inaugurate the aims and goals, and formulate the hypothesis.

This section is subdivided into three chapters and explains the related theory of the load balancing concept in Cloud-based systems. The discussion of load balancing problems with the proposed heuristic and mathematical approaches for addressing the Cloud load issues are examined in this section.

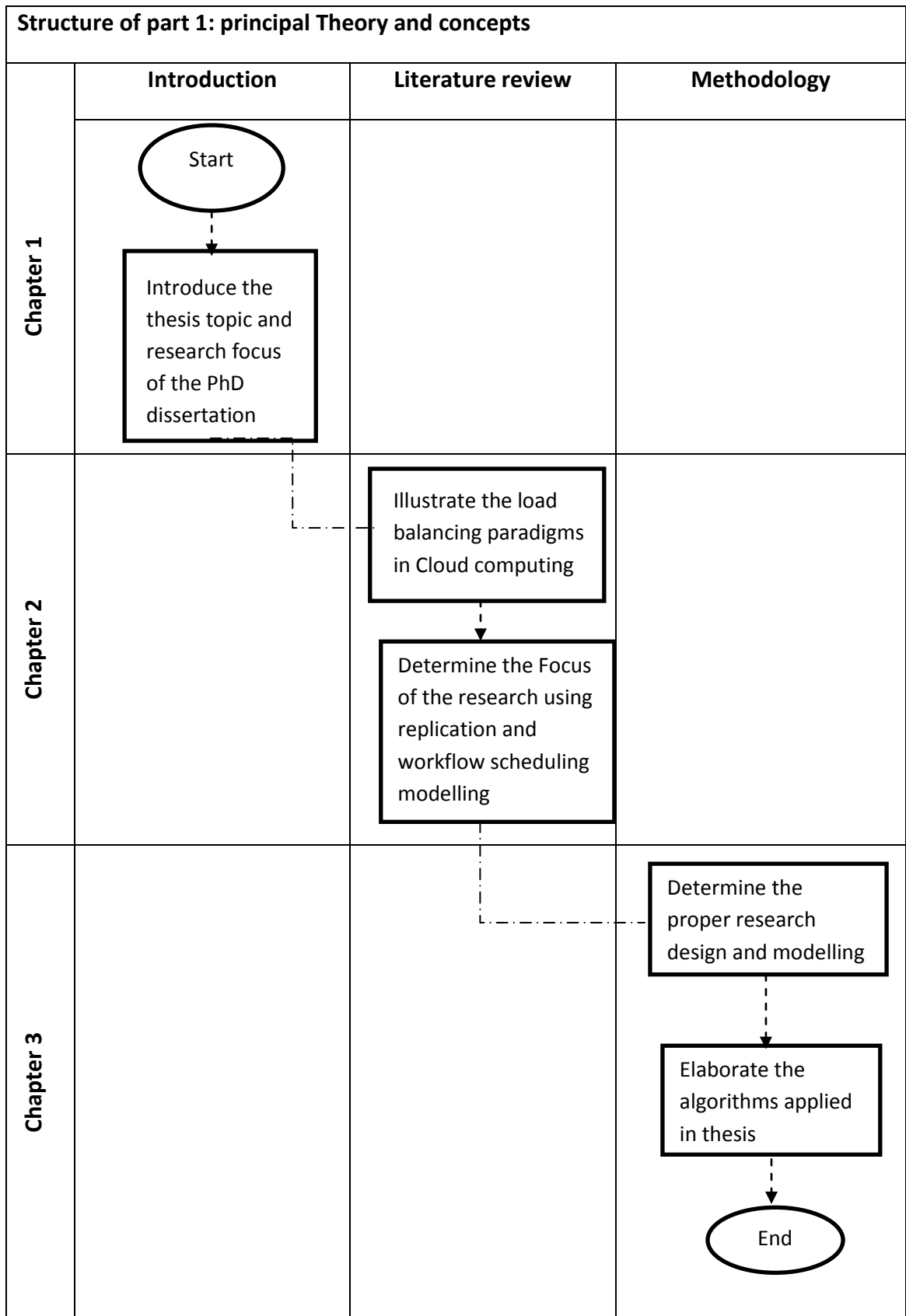
- Regulating the strategies and techniques to validate the hypothesis.

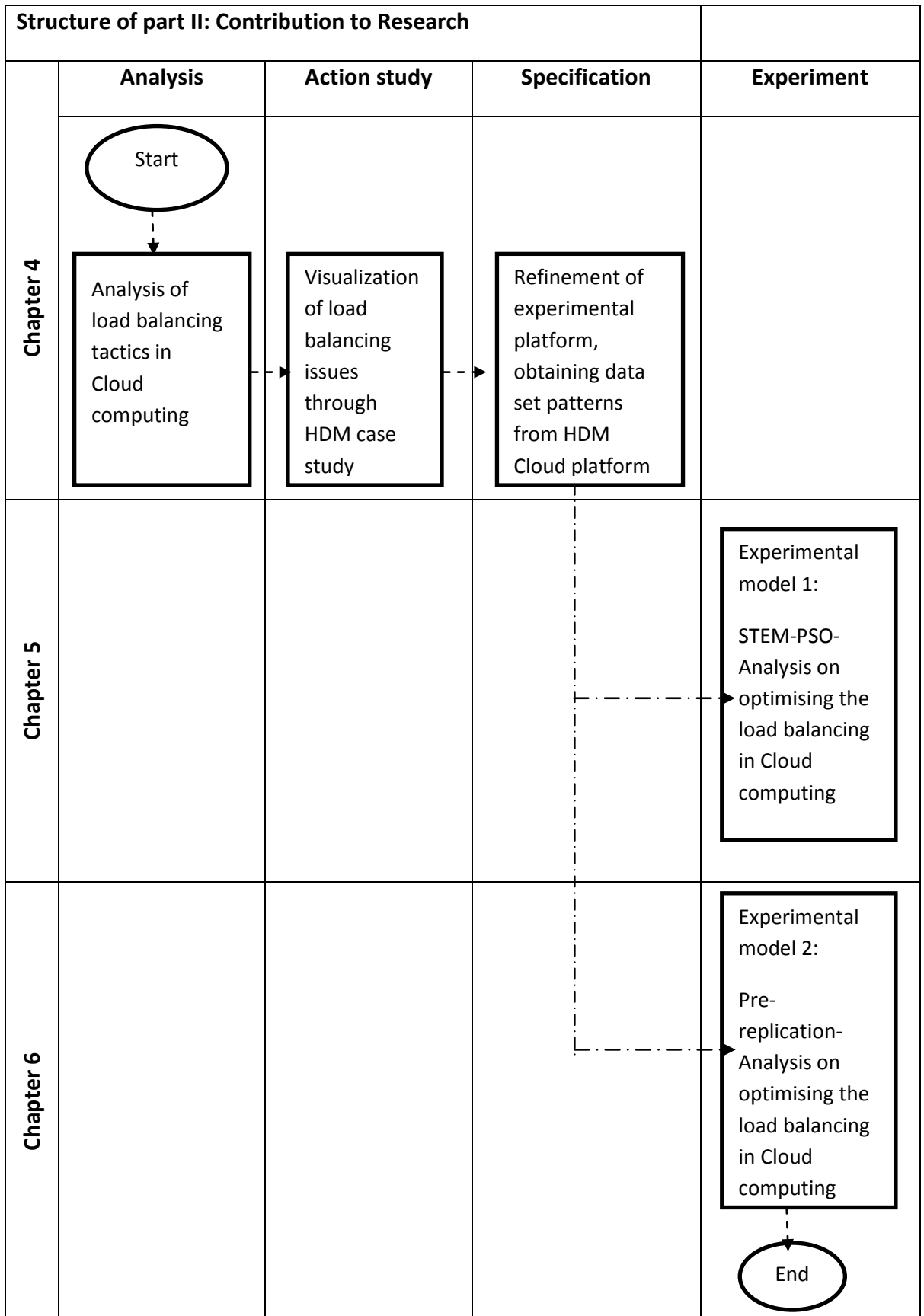
The second part covers four chapters and focuses on the research contribution which includes methodologies that explain the experimental process needed to validate the thesis aims and a research action study.

The final discussion and future work of the thesis are explained in the last chapter along with bibliography and appendices.



Table 1.1- Thesis part I and part II structure





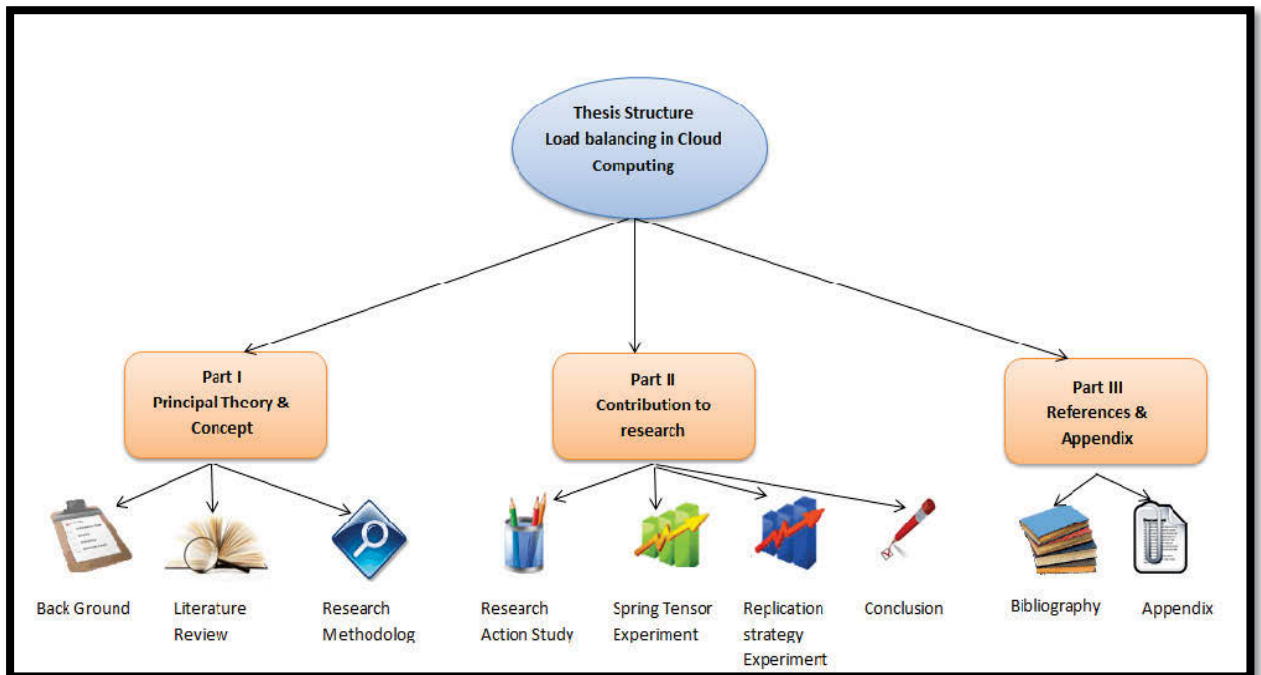


Figure 1.1- Thesis outline

Figure 1.1 presents the outline of the thesis. The details of this outline are illustrated below:

### **Part I: Principal Theory & Concept**

#### **Chapter 1:** Introduction

Pages: 3-18

Chapter 1 aims to discuss the research topic along with the research theme. The chapter explains the research questions and analyses the hypothesis. Moreover, it elaborates on the expected outcome which reflects the thesis aim to answer the research questions and validate the hypothesis.

#### **Chapter 2:** Literature Review

Pages: 19-58

The literature review provides a comprehensive review of load balancing concepts. It discusses the issues in Cloud computing along with describing the needs for designing

an optimised load balancing system that is practically important for IOT system designs and strategies.

### **Chapter 3: Research Methodology**

Pages: 59-70

Chapter 3 depicts the details of the methodologies and approaches needed to implement the observations of the experimental work. The mathematical apparatus of the heuristic approach along with the replication strategy have been examined in this chapter.

## **Part II: Contribution to Research**

### **Chapter 4: Research Action Studies**

Pages: 71-99

Based on the literature review, explained in chapter 2, the research action study establishes the strategies and technologies that could be applied in designing more optimised load balancing capability in Cloud computing. The context aims to describe the need for optimising load balancing tactics in Cloud-based systems.

### **Chapter 5: STEM-PSO Based Task Scheduling Algorithm**

Pages: 100-119

Chapter 5 is built upon the heuristic algorithm discussed in chapter 3 of this thesis. It depicts the gray-box modelling approaches with the application of Hessian matrices in the Spring Tensor Model. The chapter concludes with analytical results of the experiment.

### **Chapter 6: Replication and Load Balancing in Computing**

Pages: 120-139

The chapter describes the gray-box modelling of the replication strategy which is elaborated in chapter 3 of this thesis. The chapter concludes with the experimental results of applying the anticipatory-replication strategy in Cloud-based systems.

## **Chapter 7: Conclusion**

Pages: 140-146

The chapter concludes the major findings resulting from the experimental analysis. Moreover, it depicts the future work that could be performed based on the conducted research.

## **Part III: Bibliography and appendices:**

### **Chapter 8: Bibliography**

Pages: 148-174

Chapter 8 reports all bibliographies that have been referenced in this research.

### **Chapter 9: Appendix**

Pages: 175-189

Appendix includes the details of the experiments where could not fit within the main chapters of the dissertation.

## 1.7 Related Publications

Aslanzadeh, S. & Chaczko, Z. 2015, 'Generalized Spring Tensor Model: A New Improved Load Balancing Method in Cloud Computing', in H. Selvaraj, D. Zydek & G. Chmaj (eds), *Progress in Systems Engineering*, vol. 330, Springer International Publishing, pp. 831-5.

Aslanzadeh, S., Chaczko, Z. & Chiu, C. 2015, 'Cloud Computing—Effect of Evolutionary Algorithm on Load Balancing', in G. Borowik, Z. Chaczko, W. Jacak & T. Łuba (eds), *Computational Intelligence and Efficiency in Engineering Systems*, vol. 595, Springer International Publishing, pp. 217-25.

Chaczko, Z., Aslanzadeh, S. & Lulwah, A. 2015, 'Autonomous Model of Software Architecture for Smart Grids', in H. Selvaraj, D. Zydek & G. Chmaj (eds), *Progress in Systems Engineering*, vol. 330, Springer International Publishing, pp. 843-7.

Aslanzadeh, S., Chaczko, Z. & Chiu, C. 2014, 'Cloud Computing: The effect of generalized spring tensor algorithm on load balancing', *Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on*, pp. 5-8.

Aslanzadeh, S. & Chaczko, Z. 2013, "Generalized Spring Tensor Algorithms: with Workflow Scheduling Applications in Cloud Computing", *International Journal of Computer Applications* 84(7):15-17, Published by Foundation of Computer Science, New York, USA

Chaczko, Z., Resconi, G., Chiu, C. & Aslanzadeh, S. 2012, 'N-Body Potential Interaction as a Cost Function in the Elastic Model for SANET Cloud Computing', *Intl Journal of Electronic And Telecommunication (JET)*, 2012, Vol.58, No.1, pp. 63-70.

Chaczko, Z., Aslanzadeh, S. & Kuleff, J. 2012, 'The Artificial Immune System Approach for Smart Air-Conditioning Control', *Intl Journal of Electronic And Telecommunication (JET)*, 2012, Vol.58, No.2, pp. 193-199.

Braun, R., Chaczko, Z., Neilson, M., Nikodem, J. & Aslanzadeh, S. 2012, 'A Practical Approach for Redesigning System Engineering Processes', *International Conference on Information Technology Based Higher Education and Training, ITHET, IEEE*,

Istanbul, Turkey, 978-1-4673-2334-5/12, pp.1-8.

Chaczko, Z., Chiu, C., Aslanzadeh, S. & Dune, T. 2012, 'Sensor-Actor Network Solution for Scalable Ad-hoc Sensor Networks', Intl Journal of Electronic And Telecommunication (JET), 2012, Vol.58, No.1, pp. 55-62.

Aslanzadeh, S. & Chaczko, Z. 2012, 'The Impact of Cloud Computing on Businesses', 14th International Conference On computer aided system Theory, IEEE APCast 2012, Sydney, Australia, p.67

Chaczko, Z. & Aslanzadeh, S. 2011, 'Cloud Computing for Business: Enablers and Inhibitors', International Conference on Management Science And e-Business Engineering, ICMSBE, IEEE, Jeju Island, South Korea, pp.684-687.

Chaczko, Z., Mahadevan, V., Aslanzadeh, S. & Mcdermid, C. 2011, 'Availability and Load Balancing in Cloud Computing', International Conference on Computer and Software Modelling, ICCSM 2011, IEEE, Singapore, pp.134-140.

Chaczko, Z. & Aslanzadeh, S. 2011, 'C2EN: Anisotropic Model of Cloud Computing', 21st International Conference On System Engineering, IEEE, Las Vegas, NV, USA, pp.467-473.

Chaczko, Z., Chiu, C., Aslanzadeh, S. & Dune, T. 2011, 'Software Infrastructure for Wireless Sensor and Actuator Networks', 21st International Conference On System Engineering, IEEE, Las Vegas, NV, USA, pp.474-479.

Chaczko, Z., Aslanzadeh, S. & Klempous, R. 2011, 'Development of Software with Cloud Computing in 3TZ Team Environment', 6th International Conference on Broadband communication & Biomedical application, IEEE, Melbourne, Australia, pp.284-289.

# Chapter 2

## Literature Review

*“Successful engineering is all about understanding how things break or fail.”- Robert A. Heinlein-(1907-1988)*

The chapter reviews the past works that have been done in the field of load balancing in Cloud Computing. Figure 2.1 presents the high-level mapping of the literature review’s contents. The literature review reveals the benefits and challenges of the existing methods and reflects the possible solutions for applying more optimised load balancing techniques in Cloud-based systems. As observed in Table 1.1 section 1.6, the chapter provides the fundamental information needed for elaborating the research methodology design of the thesis.

### 2.1 Cloud Computing

It was in 1961 that McCarthy introduced a new timesharing computer technology. As an expert computer scientist, he was the first one who predicted that time sharing would lead to a more powerful computing model. He stated that in future computing power will be consumed as a public utility just like water and electricity (McCarthy 1970). His idea became popular in that time but gradually faded away in 1990. It was again at the beginning of 20th century that McCarthy’s idea resurfaced in the new form that is called Cloud Computing today (Georgi & Dalakov 2015). Since 1970, when mainframes were introduced to IT industry, computing generations have gone through dramatic changes (Lamb, J. & Cusato 1994). In early 1980, which is also known as “recessionary phase” in IT industry, personal computers appeared to increase the efficiency levels of the businesses and individual users by increasing their profitability in that period (Lazri et al. 2014). In 1990, client-server architecture offered new capabilities such as LANs to enhance the users' productivity, using the shared networks model.



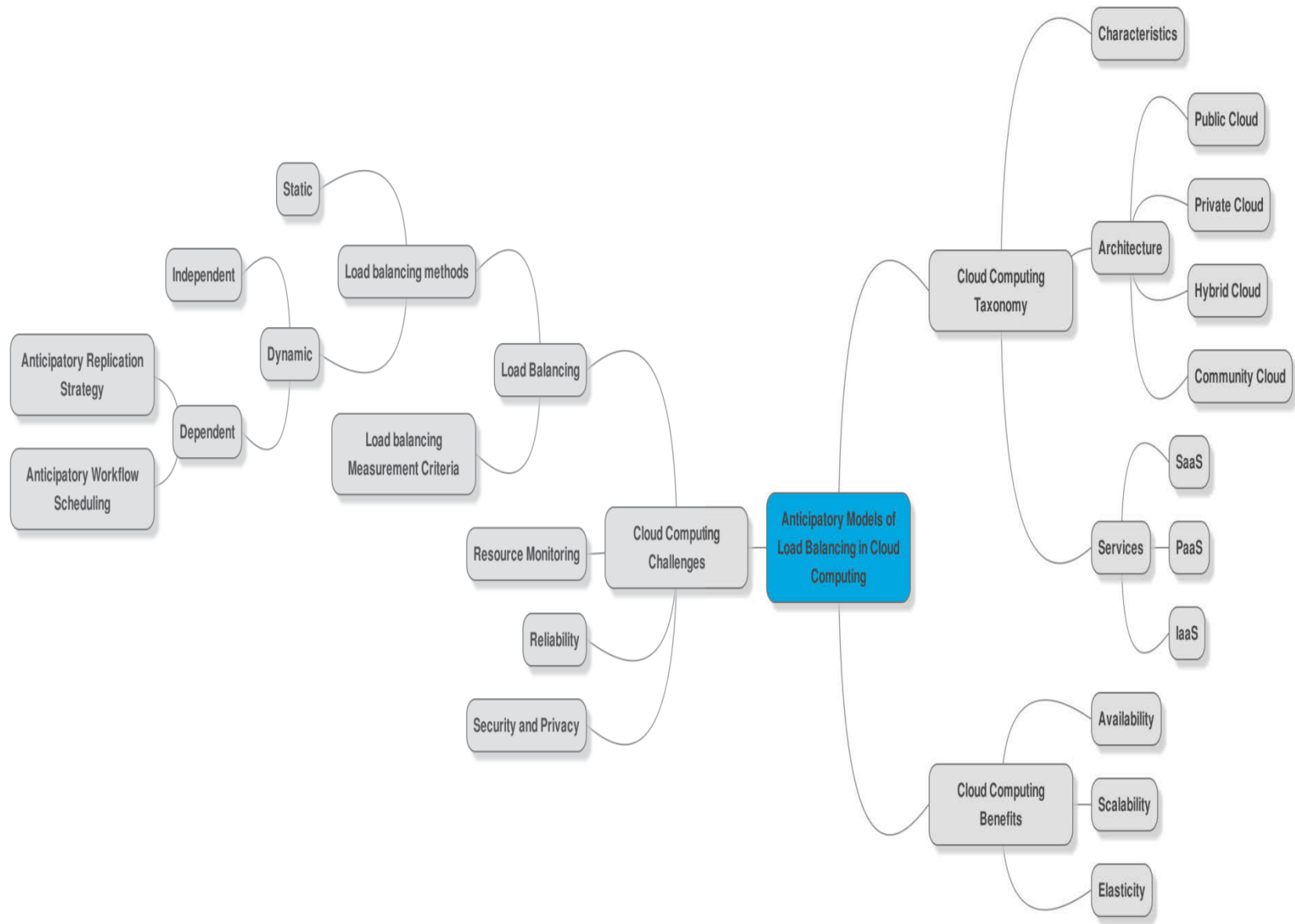


Figure 2.1- Literature review high-level mind map

Later, after the birth of the Internet, as a major innovation in IT industry, Internet Of Things (IOT) was emerged to connect users with the virtual world (Lai Gong et al. 2011). It was in 1990 that John Romkey invented a toaster that could be turned on and turned off over the Internet. It was the first device that could work over the Internet. IOT explains the structure of a virtual network which contains physical objects, controlled via the Internet. IOT saturated the users' lives by providing them with powerful devices which could sense, communicate and compute the users' need accurately and quickly (Guicheng & Bingwu 2010). Connecting a variety of things together, IOT opens a huge source of information for users and helps them with enhanced data management.

Further, IOT provides scientists with more powerful computing devices along with real-time data processing and decision making (Suresh et al. 2014).

Inspiring from IOT model, recently fifth novelty in IT industry, named as Cloud Computing, created a dramatic IT transformation by offering dynamic resource provisioning to the users. Armbust (2009) in his paper described that “*Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services.*”

Figure 2.2 is referenced by KPMG business department, depicting the IT evolution from main frames to Cloud Computing (KPMG 2011).

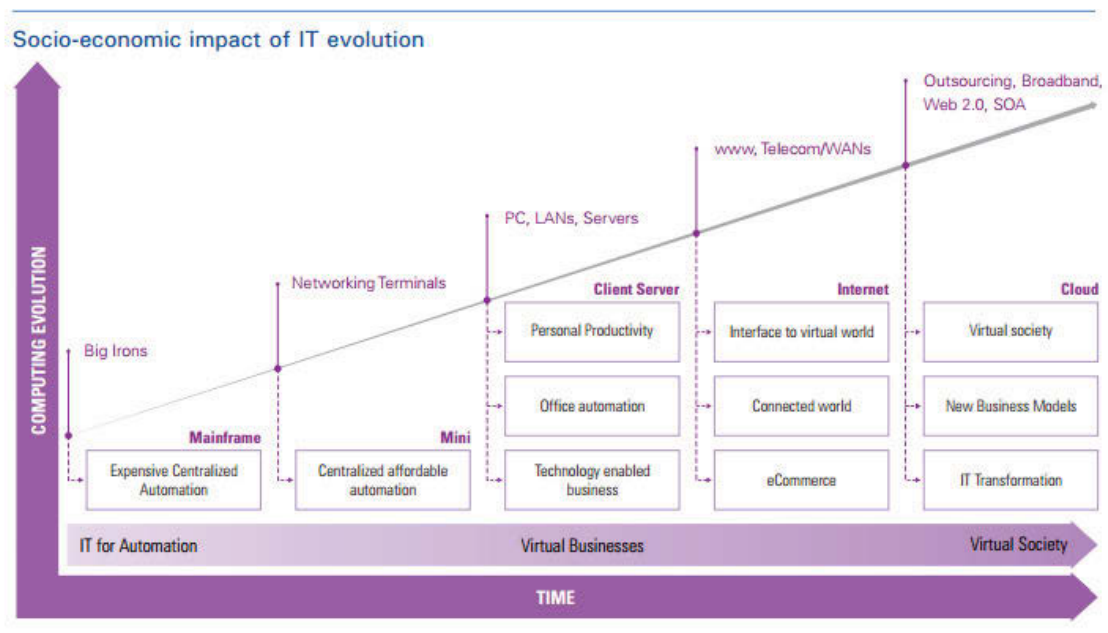


Figure 2.2- IT evolution (KPMG 2011)

Structured based on “as-a-service” paradigm, Cloud Computing promises the flexibility, scalability and cost benefits (Patel 2012).

According to Google trends, shown in Figure 2.3, Cloud Computing is a popular searching item among users. Results illustrate that since 2008, Cloud Computing became a favourite topic in comparison with Grid computing and Cluster computing.

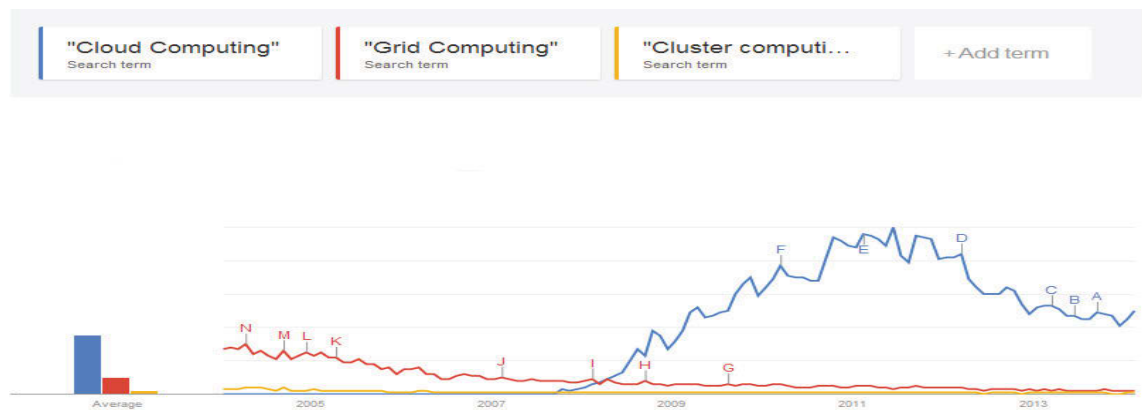


Figure 2.3- Comparing Cloud Computing, Cluster Computing and Grid Computing (Modified from Google trend)

Cloud Computing evolved from generations of grid and middleware computing Technologies. It refers to clusters of computers with the ability of dynamic provisioning in geographically distributed networks which is customizable on user’s requirements (Wilken & Colombo 2012).

Cloud Computing is one of the newly emerged innovations in IT which attracted lots of companies to replace their legacy infrastructure with newly offered technologies of Cloud (Masiyev et al. 2012). The word “Cloud Computing” is composed of two different words. “Cloud” is referring to the mesh of infrastructures, the combination of software and hardware, which provides services for end users. Efficient use of these infrastructures is ensured by managing the resources in an effective manner (Junjie, Renjie & Ke 2011).

In spite of the common characteristics, Cloud Computing has more significant advantages over grid computing. The ability of virtualisation in Cloud Computing can be considered as a prominence of that over Grid computing (Lee & Bingchiang 2011). Therefore, according to the virtualisation concept, Cloud Computing is often understood as the collaboration of scalable and elastic virtualized resources that can be provisioned dynamically over the Internet (Suciu et al. 2013).

According to Buyya (2009), Cloud Computing can be seen as “utility computing”. He noted that Cloud Computing can offer a variety of services on infrastructure, platform and software. It allows users to use its services according to their demands without any constraints. Additionally, the Cloud technology can bring profits for businesses by saving more money on their IT infrastructures.

The National Institute of Standards and Technology (NIST) defines Cloud Computing as pool of shared resources that can be configured, provisioned and released easily with minimum effort from clients and service providers (Mell & Grance 2011). The above definition highlights the pay-as-you-go model which has a key role in resource utilisation. The NIST model describes the main characteristics of the Cloud as follow (Lee, Halunga & Vulpe 2013):

- *Availability*: Users can assign and release the resources easily without long delays.
- *Accessibility*: Users can access the Cloud services by using devices such as laptop, mobile and desktops
- *Resource sharing*: Cloud is a pool of shared resources that is shared among variety of users
- *Elasticity*: Users can scale up/down their networks based on their requirements.
- *Pay-As-You-Go*: Users are charged according to their resource usage.

## 2.1.1 Cloud Characteristics

In Cloud Computing architecture qualities such as “**virtualization**”, “**availability**” and “**scalability**” are the main factors that make Cloud Computing distinctive from Cluster and Grid Computing. The details of these specifications are highlighted below:

- **Virtualization**

Virtualisation is one of the main characteristics of the Cloud Computing which enables multiple servers to run on a single device. In other words, this specification allows virtual servers to run on a hardware that brings flexibility and resource utilisation (Qingling & Varela 2011). Moreover, virtualisation reduces the power consumptions, as the number of the physical servers will be cut down. Studies show that virtualized servers consume 90% less energy than normal physical servers (Salapura 2012).

- **Availability**

Availability highlights the smart way of resource provisioning that attracts most of the businesses to change their legacy infrastructure with Cloud Computing.

Cloud Computing enables the newly built entrepreneurs to grow faster by saving costs on available IT infrastructures and focusing on their core business elements (Gonzalez & Helvik 2012). Considering all these benefits, it is almost effortless for organisations to start their businesses without using the Cloud Computing facilities.

- **Scalability**

Scalability provides the organisations with the required services within the enterprises. Depending on the workload of the organisations, the resources can be under-provisioned or over-provisioned (Yoo & Dong 2010). Cloud Computing can control the resource requirements by scalability specifications which help organisations to save a huge percentage of their budget on IT infrastructure.

## 2.1.2 Cloud Taxonomy

Armbrust (2009) suggests that Cloud Computing can be categorised based on its capability and accessibility. Considering the accessibility concept, four types of Cloud Computing have been offered to Cloud users, Figure 2.4 summarises these categories (Conres 2014).

- **Public Cloud:** Public Cloud enables end-users to benefit from a variety of services and resources over the Internet. Most of the services in public Cloud are free, but in some cases, users may be charged according to the “pay-as-you-go” model (Wang & Xu 2008).

Employing public Cloud, organisations can adapt to unlimited scalability opportunity which allows scaling their shared resources according to the requirements (Xu et al. 2013).

- **Private Cloud:** Private Cloud provides an infrastructure which is solely designed for specific organisations. In this type of Cloud, all the resources and services are private for that management system. Accordingly, organisations may use the dedicated hardware which cannot be shared with other parties (Hongli et al. 2013).

It has been argued that private Cloud is demolishing the real meaning of the Cloud Computing as sharing; multi-tenancy and scalability are missing in this type of Cloud (Prabavathy et al. 2013).

- **Hybrid Cloud:** Hybrid Cloud is described as combination of the infrastructures that organisations own, along with the opportunity of using the public Cloud benefits when needed (Breiter & Naik 2013). For example, an organisation can store its critical and secure data on private Cloud that costs more but, on the other hand, employs public Cloud to benefit from its free services (Naik et al. 2014).
- **Community Cloud:** Community Cloud infrastructure will be shared among organisations from the same community where the users are less than the

number of the public Cloud end-users but more than the number of the private Cloud clients (Selimi et al. 2014).

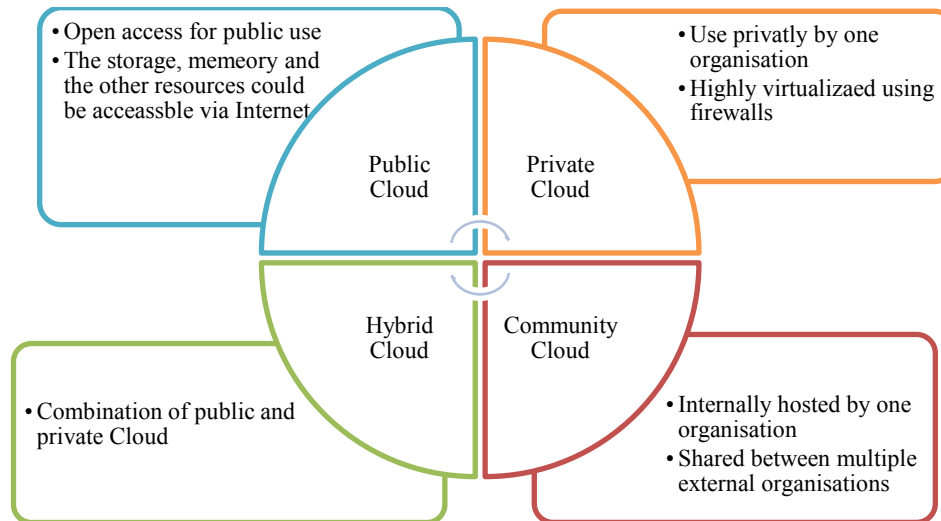


Figure 2.4 -Cloud taxonomy

### 2.1.3 Cloud Architecture

According to the reference architecture of Cloud Computing, Figure 2.5, Cloud Computing is composed of 5 main components (Buya et al. 2008), (Bojanova et al. 2011), (Chandrane et al. 2010).

- Cloud consumer: are the main stakeholders of the Cloud Computing services who may be billed according to their resource usage.
- Cloud auditor: is mainly focusing on validating the security, privacy and performance specifications of the Cloud systems.
- Cloud provider: makes the services available to consumers. Cloud provider has the responsibility of managing the technical infrastructure, provisioning the resources and ensuring the privacy and security of the Cloud.

- Cloud broker: is the intermediate connection between Cloud consumer and Cloud providers.
- Cloud carrier provides the Cloud consumers with the proper access to Cloud services using the network and telecommunication facilities.

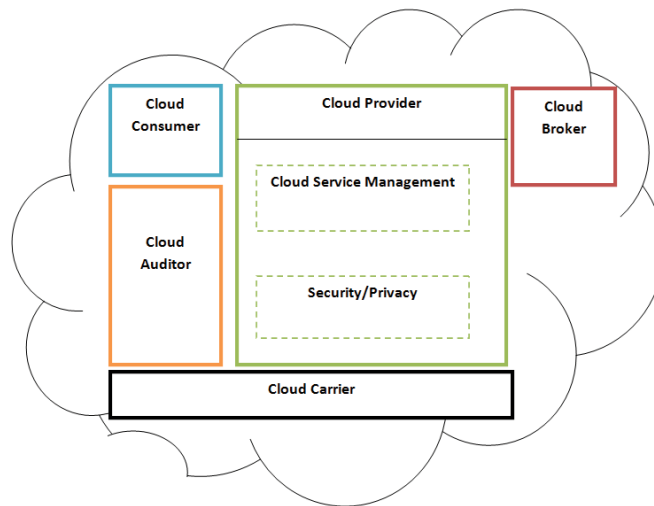


Figure 2.5 -Reference model of Cloud architecture, adopted from, (Buya et al. 2008)

From the above-mentioned components, it can be seen that Cloud provider component plays a critical role as it is providing users with management services (Yanmei et al. 2011). Resource provisioning and task scheduling can be managed with Cloud provider components. In Cloud Computing, a task is defined as a minimum unit that should be allocated on available resources. A job or meta-task also is referred to the sets of tasks that could be considered for scheduling purposes (Annette et al. 2013). As it is shown in Figure 2.5, Cloud provider and Cloud broker are working together to schedule the tasks on proper resources.

The broker then is responsible for scheduling the jobs on recognised and available resources. Also broker should always be aware of the status and availability of the resources. Within the Cloud provider component, there is a segment named as “Cloud stack” which is described in more details in the next section.



## 2.1.4 Cloud Stack

Cloud Computing is composed of a different range of services, Figure 2.6, that are built on top of another. These services include:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

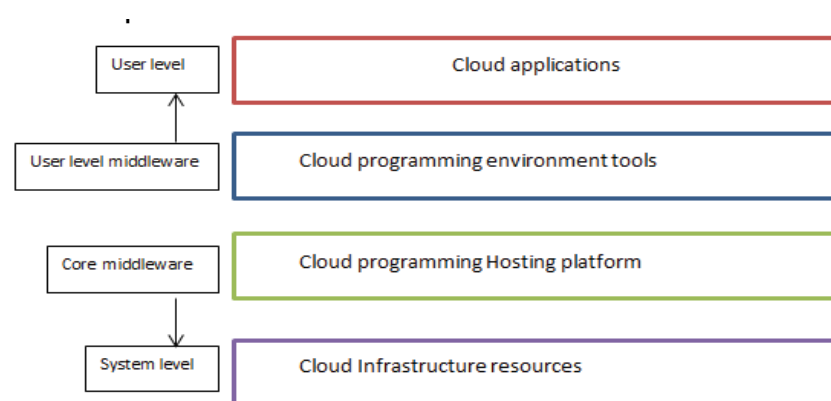


Figure 2.6- Cloud stack categorisation, adopted from (Barkat et al. 2014)

### 2.1.4.1 Infrastructure as a Service (IaaS)

IaaS provides users with the pool of physical and virtual resources such as servers, computers, hypervisors and VMwares (Annette et al. 2013). Applying the virtualisation technique, IaaS offers scalability along with strong computation power and more storage capacity (Wei-Tek, Xin & Balasooriya 2010).

IaaS offers different resources such as storage, network and operating systems on demand. This helps clients save their budget on buying the infrastructure that may/may not be used in future within their networks (Metwally, Jarray & Karmouch 2015).

It should be mentioned that IaaS can be accessed via public or private Cloud. Public Cloud allows users to assign/ release resources on a self-service manner. However, in private Cloud only users who access the private network are allowed to use the offered services (Kozlovsky et al. 2013). Hybrid IaaS also refers to the combination of the

private and public Cloud. Hybrid IaaS, in this case, could be more beneficial as it helps users to save more by benefiting from public shared resources (Tang & Chen 2014).

### **2.1.4.1.1 Characteristics of IaaS**

NIST highlights the main characteristic of IaaS as follow (Mell & Grance 2011):

- Different resources such as network and storage are shared as a service.
- With virtualisation techniques, multiple users can use a single physical hardware.
- Resources can scale up/down dynamically.

Applying IaaS will be useful if (Kozlovsky et al. 2013):

- Scalability is essential for organisation's networks.
- The organisation is newly built and is growing rapidly, so investing on physical hardware will be costly.
- Companies want to minimise their administrative costs and focus more on their operating investments.

### **2.1.4.2 Platform as a Service (PaaS)**

PaaS helps users to create their customised software. It is also known as “Cloud ware” which provides a development platform; assists users to design, develop, implement and test their services on Cloud. Google App engines and Microsoft Azure are popular examples of PaaS (Wenboet al. 2012).

In Cloud Computing scheduling decision should be made in shortest time as there are many competitors competing for the available resources. These services can be easily provided by PaaS. (Abraham 2000).

## **2.1.4.2.1 Characteristics of PaaS**

The main characteristics of the PaaS include (Mell & Grance 2011):

- PaaS has multi-tenant architecture. In this way, multiple users can work on the process of applications development simultaneously.
- PaaS offers services such as load balancing, utilised task scheduling and failover techniques.
- PaaS plays a key role in software development phase. It provides developers with workflow management system which is independent of the applied data source (Gopularam, Yogeeshha & Periasamy 2012).
- PaaS is useful for the latest agile software development techniques where multiple developers wish to work on the same development project with an automated testing procedure.

## **2.1.4.3 Software as a Service (SaaS)**

SaaS provides users with on-demand services. Usually, it deploys on users browsers and does not need to be installed and maintained individually (Yang et al. 2010).

The software is automatically updated and can be used by multiple users without buying an additional license. Salesforce and Google maps are examples of this category (Hong 2010).

### **2.1.4.3.1 Characteristics of SaaS**

Some of the main characteristics of SaaS can be highlighted as (Gibson et al. 2012):

- Online access to different profitable software.
- Using SaaS services, companies do not need to buy different licences for their users.
- SaaS is delivered in a one-to-many manner.
- SaaS upgrades software versions and handles the software issues automatically.

## **2.2 Cloud Computing: Challenges and Benefits**

Cloud Computing offers plenty of benefits to the users. Infinite availability of the resources, beneficial payment model, scalability and elasticity specifications of the Cloud networks motivated businesses to change their legacy system with the novel Cloud infrastructure (Armburts 2009).

Scalability and elasticity can be highlighted as the main benefits of the Cloud Computing. Scalability enables users to add or remove resources from their networks at any time. Conversely, elasticity emphasizes on dynamic resource allocations. In fact, elasticity explains the definition of the "pay-as-you-go" model which is allocating and releasing the resources according to the user's requirements (Galante & de Bona 2012). However; Shameem (2013) argues that, although elasticity introduces many benefits to the users, effective load management is essential to guarantee the availability of the resources. Table 2.1 summarises the main benefits of the Cloud Computing:

Table 2.1- Benefits of the Cloud Computing

<b>Benefit</b>	<b>Opportunity</b>
Scalability	Add or remove resources on demands
Elasticity	Allocate and release resources upon usage
Mobility	Accessing the Cloud network is not dependent on time and location
Low infrastructure cost	Helps small businesses to grow sooner
Latest updates in IT	Regular updates of the system, implemented by Cloud providers
Increased data storage	Access to large storage capacity for data storage and backup plan
Disaster recovery	Using the virtual backups; recovery will be 4 times faster than using another system than Cloud
Availability	Rapid deployment of the infrastructure to access the resources
Billing and payment	Users will be charged per services based on their usage

Besides all the above-described benefits, Cloud Computing should be able to address the different obstacles to ensure an efficient deployment of an elastic, scalable, secure and reliable platform (Moreno-Vozmediano, Montero & Llorente 2013). Table 2.2 highlights the main issues in the current Cloud systems:

Table 2.2 -Challenges in the Cloud Computing

<b>Challenges</b>	<b>Opportunity</b>
Interoperability	Lack of standards for service portability between Cloud providers
Security and Privacy	Lack of improved techniques in authorisation and authentication for accessing the users information
Resiliency	The ability of the system to provide users with standard level of services while experiencing faults and challenges in the system
Reliability	The chance of failure in standard period of time
Energy saving	Defining a standard metric for effective power usage and an efficient standard of infrastructure usage
Resource Monitoring	Lack of accurate monitoring mechanism using sensors to collect the data from CPU load, memory load and etc.
Load Balancing	Lack of standard way of load monitoring and load management for different Cloud applications

Table 2.2 highlighted load balancing as one of the open challenges in Cloud Computing.

As Cloud Computing has multi-tenancy structure, availability and efficiency of the resources should be considered as essential foundations of the Cloud architecture that can be guaranteed by proper load balancing method (Domanal & Reddy 2014).

Recent studies showed that, optimised Cloud Computing could be seen as an elastic network of resources that are interacting with each other to minimise the waiting time and utilise the throughput (Ganget al. 2012). Therefore, load balancing and resource management can be highlighted as one of main concerns in Cloud Computing as they are impacting the network performance directly.

Next sections will discuss load balancing issues in Cloud Computing in more details.

## **2.3 Load Balancing in Cloud Computing**

### **2.3.1 Load Balancing: Concept, Definition, Benefits, Challenges**

The previous sections reviewed the main theories of Cloud Computing. Considering the main challenges, load balancing is highlighted as a major characteristic of Cloud Computing which is still regarded as an open area for research.

As featured in Cloud taxonomy, load balancing can play an important role in public, private, and hybrid Cloud. Therefore designing an optimised load balancer either in infrastructure or platform level can optimise the Cloud performance and provide users with enhanced quality of service (Maarouf, Marzouk & Haqiq 2014).

Load balancing is one of the main concerns in Cloud Computing that has a major impact on defining the availability of the resources (Mathur & Nishchal 2010). "Availability" refers to the ubiquitousness of the network information but was always the main burden in Cloud-based systems (Gupta et al. 2013).

Load balancing plays a key role in optimising the network performance. By proper load balancing strategy, tasks could be scheduled evenly among available resources which could lead to a balanced network (Karim, Chen & Miri 2015). The lack of efficient load balancing tool can result in experiencing a long access time for users. Today with emerging the new techniques, the service providers are trying to apply the automated load balancing algorithms to promote the availability and performance of the Cloud systems (Randles et al. 2010).

Effective load balancing algorithms include the following elements (Kruber, Hogqvist & Schutt 2011)

- Tasks will be distributed evenly between available resources.
- Makespan will be minimised.
- Resources will have maximum utilisation.
- Resource availability will increase.
- Network performance will improve.

## 2.3.2 Load Balancing Measurement Criteria

Davein (2005) suggested that load balancing algorithms should be robust enough so they can be compatible with variety types of applications (Devine et al. 2005).

To design more efficient load balancing algorithms, the following checklists are suggested (Chiu, Raghavendra & Ng 1989), (Ming & Zu-Kuan 2010), (Pius & Suresh 2015), (Domanal & Reddy 2014).

- Complexity: The algorithm should be smart and simple. Complexity can apply more overhead on an algorithm. Therefore, more resources should be engaged while algorithm is being executed.
- Scalability: The algorithm should be adaptable to the network expansion and abridgement.
- Fault tolerance: The algorithm should be smart enough to handle the load if any failure happens during the run time.
- Performance and makespan: The load balancing algorithm should optimise the Cloud performance by minimising the total execution time



## 2.3.3 Load Balancing Methods

In general load balancing algorithms are categorised into two main groups (Figure 2.7) as follows:

- **Static load balancing:** In a static environment, prior information about node capacity, processing power, memory and performance is needed.

The statistics requirements cannot be changed at run-time. Although the static environment is much easier for implementing the load balancing algorithms, it is not suitable for heterogeneous computation models (Ruixia & Xiongfeng 2010).

A simple example of a static algorithm is Round-Robin algorithm. In this resource scheduling method, the task that comes first will be served first and based on the time sharing manner the resource that is least loaded will be allocated to complete the tasks (Zhenzhong et al. 2013).

- **Dynamic load balancing:** In this environment despite the need for prior, like static environment, the algorithms operate according to the run-time statistics (Wenhong et al. 2011). These load balancing algorithms are more flexible to change and they are highly adaptable to Cloud environments.

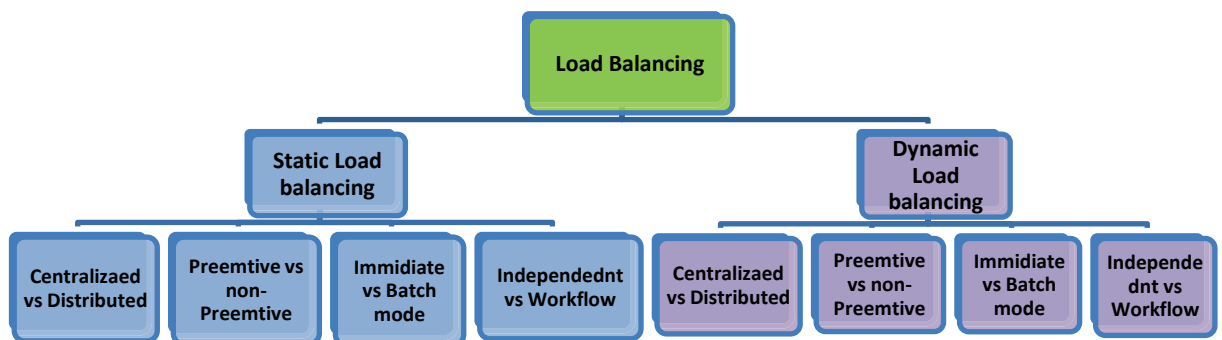


Figure 2.7 -Load balancing categories

Each of the static or dynamic algorithms could be divided into 4 different categories:

- *Centralized vs Distributed*: Centralized model is designed in a way that it has a central controller and can store the information of all the resources in a centralised way. However; the model is not adaptable in terms of network scalability (Venkatesan & Solomi 2011).  
On the other hand, although distributed load balancer does not have any centralised controller but it can perform well in a case of scalability and fault tolerance, so it can support elasticity (Vijindra & Shenai 2012).
- *preemptive algorithm vs non-preemptive*: Preemptive algorithms allow interruptions during algorithm run time. As an example, depending on the tasks priorities, the algorithm can stop to change the order of the tasks in its queue (Shobana, Geetha & Suganthe 2014). On the other hand, no interruptions are allowed in non-preemptive algorithms until all assigned tasks are scheduled on the available resources (Shameem& Shaji 2013).
- *Immediate vs batch mode*: In immediate mode, as soon as the tasks are assigned for scheduling, they will be sent to the queue. In batch mode, tasks should be grouped based on the criteria then the group of tasks will be sent to the scheduling queue (Donge-sheng et al. 2004).
- *Independent vs workflow*: In workflow modelling the dependencies between tasks should be investigated before they could be assigned on available resources (Luo et al. 2012). DAG graphs and Petri nets are the common languages that can represent the workflow scheduling algorithms.  
Independent modelling, however, will schedule the tasks without considering their interconnectivity (Zhangjun et al. 2010).

## **2.4 Focus of This Research**

As previously described various load balancing techniques in Cloud find lots of interests among members of the research community.

As the focus of this research is on dynamic and specifically on dependent and independent load balancing algorithms, the rest of this chapter is examining the various categories of these computational models. The aim is to explore both benefits and challenges of the traditional approaches as well as new developments in the field.

### **2.4.1 Load Balancing Algorithms: Independent vs Dependent**

Independent and dependent scheduling methods are considered as the key approach in terms of load management. Dependent scheduling, however, is attracting more attentions (Singh 2013).

Dependent scheduling is suitable for those types of tasks with dependent structured patterns. In these models, each job is composed of several dependent tasks. Therefore, execution of one task is dependent on another (Tilak & Patil 2012).

Unlike the independent tasks, failures in dependent-tasks execution affect the performance of the whole system. Currently, there is a lack of effective algorithm that deals with load balancing on dependent structured jobs (Zhang & De Sterck 2012).

Therefore, to understand the main gaps, in this research a comprehensive literature review has been done on more popular dependent and independent load balancing algorithms.

## 2.4.1.1 Dependent Load Balancing Algorithms

Dependent scheduling algorithms are categorised as NP-hard problems which means that there is no standard algorithm that can generate an optimal solution for all the scheduling problems (Greenwood 2001). NP is an acronym for the non-deterministic polynomial. Chapman (1994) in his paper noted that “NP-complete is a class of problems that has no known efficient algorithm for finding an optimal solution.” In other words, there is no algorithm that can cover the increased steps of the problem with the polynomial rate. The NP-hard problem is considered as a subset of NP problems that cannot be resolved with non-deterministic machines (Bernard & Graham 1989).

Dependent scheduling is referring to mapping and managing the inter-dependent tasks on available resources. Dependent algorithms are mainly designed to help the Cloud infrastructure for supporting the large distributed collaborative e-business and e-science applications (Bala 2011). Comparing with e-Business functions, e-science applications are more data intensive and need large and complex processing methodologies. Moreover, they are dynamic and need high-performance infrastructures (Jenn-Wei, Chien-Hung & Chang 2013). In his work, Liu (2013) categorised the dependent scheduling algorithms into two groups:

- Deterministic
- Non-Deterministic

With deterministic algorithms, the dependencies of the tasks can be recognised at the start time. However, in non-deterministic algorithms, the dependencies between tasks and I/O data will be perceived at the run time (Amoivalis, Tsili & Kladas 2012).

Additionally, dependent scheduling algorithms can be classified into:

- Best-efforts (heuristic) scheduling algorithms.
- The quality of service (QOS) constraint scheduling algorithms.

The best effort algorithms are trying to minimise the execution time. But they are not compatible with pay-as-you-go billing model (Peng & Ravindran 2004).

On the other hand, QOS-based algorithms are considered for high quality applications. In this type, tasks will be grouped according to their quality requirements (Liu 2013).

Buyya et al (2008) also divided the dependent scheduling algorithms into two levels:

- Functional and non-functional service level scheduling: Analysing the submitted tasks, the global broker will verify the functional and non-functional requirements such as quality, time and cost. The broker will use this information to offer a suitable service for scheduling the tasks on available resources. Generally the structure of this method is static, and will be executed upon the runtime.
- Task-level scheduling: This methodology is mainly investigating the precedence of each task before execution. With this method, optimal scheduling plan will be selected. Task-level scheduling is static and does not need to be implemented regularly.

Dependent applications can be scheduled with a variety of heuristics algorithms such as min-min-, max-min, FCFS. However, they need to accurately manage systems by using more complex scheduling policies in order to satisfy their QOS requirements (Zuo & Lin 2010).

Therefore, to highlight the main gaps in dependent scheduling methods, we selected the following major workflow load balancing algorithms (Table 2.4) and compared them as follows:

- Heterogeneous Earliest Finish Time (HEFT): In this method, tasks will be ranked based on their execution time. Tasks with lower execution time will gain the highest priority for resource scheduling (Bala et al. 2011). Although the resource utilisation is inevitable with this method but the low makespan is a major challenge within this methodology (Sakellariou & Henan 2004).

- Scalable Heterogeneous Earliest Finish Time (SHEFT): SHEFT is considered as an extension to HEFT methodology described above. In this method, the earliest finish time and start time for each task will be calculated. The task with minimum execution time will be allocated on the first available resource. Moreover, each resource should have a finishing time (Chopra & Singh 2013). The earliest finishing time of the allocated task should be earlier than minimum finishing time of the available resource. In this case, if any of the resources are kept idle more than a defined threshold then they can be released based on their timing history (Cui & Shiyong 2011).
- Transaction intensive cost-constrained: Similar to market-oriented workflow algorithms, upon scheduling the tasks on available resources costing and timing will be considered in this algorithm. The overall goal in this algorithm is to minimise the execution time under the considered deadline(Yun et al. 2008).
- QoS heuristic workflow scheduling (An optimal workflow based scheduling): The goal of this algorithm is on CPU utilisation. In this algorithm, each task in a workflow will be analysed according to their start time, finishing time, favourite processor and favourite predecessor. Then the workflow will produce sub-tasks for calculated parameters. Finally, based on the resource monitoring factors, such as time, cost and reliability, tasks will be mapped on available resources. If the idle resource is not available, tasks will be compacted (Varalakshmi et al. 2011).
- A revised discrete particle swarm optimisation: This algorithm considers the discrete characteristic of PSO to optimise the makespan while minimising the associated costs. In this algorithm, each particle will learn their best position locally and globally. The condition for each movement could be defined with different QoS elements such as deadline, budget or data transfer rate (Nguyen & Mengjie 2014). Due to the discrete property of scheduling, each particle will learn from its previous position which should be the highest value in global position (gbest). Then all the unmapped tasks will be chosen from other

particles and finally the gbest of the particle will be the optimal order for the task (Zhangjun et al. 2010).

- **Deadline constrained workflow scheduling in software as a service:** This algorithm mainly is trying to minimise the execution time by meeting the deadlines based on QoS requirements. In this algorithm, workflow tasks will be scheduled by following the concept of the partial critical path. The algorithm will use recursive scheduling, using the path that tasks were previously scheduled (Abrishami & Naghibzadeh 2012).
- **Deadline and budget constraint:** Most of the scheduling algorithms are trying to schedule the tasks on budget and time. But it is hard to satisfy these two requirements together. Therefore, Wang et al (2011) suggested a new method which is adding a new metric named as relative cost (RC) to the algorithm. In their method, resources are assumed heterogeneous and will be ordered on their time-cost. Tasks will be allocated on the first available resource where cost and time are less or equal to the remaining budget and deadline.
- **Dynamic critical path:** This algorithm will determine the efficient mapping of the tasks based on the "Dynamic Critical Path" definitions. In this algorithm, tasks will be prioritised on their estimated completion time. Scalability and fault tolerance is not considered in this algorithm (Rahman et al. 2007).
- **A practical swarm optimisation based heuristic for workflow scheduling:** In this heuristic algorithm computation cost and data transmission cost should be considered. Each task has its own communication cost, which could be considered as a weight for each task. Using this methodology, the workflow makespan will be minimised (Pandey et al. 2010).
- **Genetic Algorithm:** GA algorithm is one of the evolutionary algorithms which is inspired by evolutionary biology. In this algorithm, the solutions are shown with strings known as a chromosome. Three main operations such as selection,

crossover, and mutation will be conducted on each group of chromosomes. This algorithm is mainly focusing on minimising the makespan and costs (Sawant 2011).

Table 2.3 summarises the details of the above-mentioned algorithms.



Table 2.3- Summary of the reviewed dependent scheduling algorithms

Load balancing algorithm	Static/Dynamic	Benefits	Challenges
HEFT	Static	Makespan will be improved	Scalability and resource utilisations are not considered within this algorithm.
SHEFT	Dynamic	Optimise execution time	Overhead will be added to pre-calculating the start/finish time for the tasks
Transaction-intensive cost constrained	Static	Minimise cost based on considered time	Only will consider the in-time task scheduling processes
QoS heuristic workflow scheduling	Static	CPU utilisation is achieved through applying different parameters of QoS	Trustworthiness and fault tolerance are not considered in this algorithm.
A practical swarm optimisation-based heuristic	Dynamic	Makespan will be minimised.	Scalability and fault tolerance is not considered.
Deadline Constrained workflow scheduling in software as a service	Dynamic	Minimise cost while meeting the deadlines. Minimise makespan.	Complex algorithm. Computation cost of the algorithm is high
Deadline and budget constraint:	Static	Suitable for large scale distributed systems.	Do not consider communication and execution cost of the tasks
Dynamic Critical Path	Dynamic	Tasks will be prioritised base on their completion time	Do not consider scalability and fault tolerance
Discrete particle swarm optimisation	Dynamic	Minimise cost Better performance on makespan	Fault tolerance is not considered in this algorithm.
Genetic Algorithm		Minimising the makespan and costs	Fault tolerance is not considered

## 2.4.1.2 Independent Load Balancing Algorithms

Below, the main independent load balancing algorithms are defined. Table 2.4 includes the benefits and challenges of the explored algorithms.

- Round robin: This algorithm is considered as a static load balancing algorithm. Round robin operates based on the time spans assigned to each node. With this algorithm traffic will be distributed evenly. However, as the algorithm is static, it cannot manage the network's load in a real-time manner (Radojevic & Zagar 2011).
- Dynamic round robin: This algorithm is an improved model of Round Robin algorithm. In this model, the tasks execution sequence will be recorded automatically based on the current status of the network. The algorithm is creating less overhead than Round-Robin algorithm and it can improve the response time. However, it has low performance in busy environments (Batcher & Walker 2008).
- Signature patterning: This algorithm works with different time-slots. It captures signatures from executed tasks and resources to make patterns. The pattern will show the precise resource status and it has a certain threshold. If the patterns show that the resource reached its threshold, based on the captured signatures, the load will be distributed on less overloaded nodes (Ramaswamy 2009).
- Task consolidation algorithm: The algorithm will operate dynamically according to the heuristic methodologies. Different components will be considered for the task allocations. Eventually tasks will be assigned on resources that are most cost effective and more energy efficient. This algorithm is only suitable for local Clouds and cannot support large scale networks (Gaikwad-patil 2012).

- Dynamic replication algorithm: In this method, algorithm will try to replicate the files on local servers. In this case, when users want to access a file, they could find it on the local server. Using this method the access time would be minimised (Gaikwad-patil 2012).
- Map Reduce: This algorithm has two main functionalities:  
 Firstly, all the jobs will be divided (Mapped) into subtasks. Each subtask will have its own key ID. In next step, all the IDs would be converted to hash keys. Then, the reduce function will do an operational summary on each sub-groups with their hash IDs and it will generate a single output (Nurain et al. 2012).  
 In final stage, a central node is dedicated to compare all the generated single outputs and assign them on available resources. The only problem with this algorithm is related to its overhead. As mapping and reduction step can be done in parallel, nodes might be overloaded (Dongjin & Kwang-Mong 2012).
- Ant colony: In this algorithm, ants will move forward to find the first under-loaded resource. If the ant finds the first under-loaded server, it will move forward to check the next server status. If the next resource is under-loaded as well, it will move forward to find the last under-loaded resource. Otherwise, the ant will move backward from overloaded node to the previously available resource (Kun et al. 2011).
- Index name server: The algorithm will minimise the data replication and data redundancy. Based on the maximum transition time and bandwidth, the algorithm will do some calculations to find an optimum server for task scheduling. In this algorithm, the connection weight between server and nodes will be calculated to clarify whether the server can handle more nodes or not (Swarnkar 2013).
- Min-Min: In this algorithm, minimum completion time for each task will be calculated. Then the task with the overall minimum completion time will be

selected. The same iteration will happen until all the tasks are allocated on available resources (Shu-Ching et al. 2010).

- Max-Min: In this algorithm, minimum completion time for each task will be calculated. Then the task with maximum completion time will be assigned to the first available resource. The iteration will continue until all tasks are assigned on the available resources (Manish and Cheema 2012).

- Artificial bee colony: This algorithm is designed according to the behaviour of the honey bees.

In this model three main components exist: Employed honey bees, un-employed honey bees and food sources. Bees will start looking for the food source randomly and they will record the information related to the location of the source and its profits. Then they will start gathering honey and returning back to their hives (He & Jia 2012). At this stage, bees can get back to the same source to gather the remaining honey or they can remain idle.

In terms of load balancing when requests are submitted to the load balancer, at first its profit should be analysed. Generally the CPU time and waiting time will be measured for this matter. This information will be recorded in the database. Then if beneficial, tasks will be assigned on selected resource (Haozheng et al. 2012).

- The discrete version of PSO (DPSO): PSO is a type of evolutionary algorithm that works base on the velocity and portion of the particles. Each particle has a local memory that memorises its velocity and position. Additionally the particle can learn from its adjusted velocity. In this case by each movement based on the velocity location, limited between (-1, 0, 1), the position of the particle will be updated. If the  $v_t = -1$  , particle will learn the new position from its neighbour, if  $v_t = 0$ , there will be no change in the position and if  $v_t = 1$ , particle will learn from its past movement (Pandey et al. 2010).

Table 2.4- Summary of the reviewed independent algorithm

Load balancing algorithm	Static/Dynamic	Benefits	Challenges
Round Robin	Static	Distribute the traffic evenly based on time slices	Real-time load cannot be considered Longer waiting time
Dynamic Round Robin	Dynamic	Minimise the waiting time Minimise response time	The performance of the algorithm is low
Signature patterning	Dynamic	Resource status and allocation are managed precisely	Extra overhead will be added due to pattern capturing and comparison
Task Consolidation algorithm	Dynamic	Cost effective Energy efficient	It could be applied on local Clouds
Replication algorithm	Dynamic	Improve access time	Does not support pre-replication and fault tolerance
Map Reduce	Dynamic	Suitable for large distributed networks	Due to parallel processing, nodes can be overloaded
Ant Colony	Dynamic	It uses meta-heuristic approach	In-effective resource utilisation is needed
Index Name Server	Dynamic	Minimise the waiting time Improved fault tolerance plan	Extra overhead will be added on servers due to connectivity calculation
Min-Min		Improve availability	Cannot support fault tolerance
Max-Min		Minimise the waiting time	Cannot support fault tolerance
Artificial bee Colony	Dynamic	Suitable for scalability	System throughput cannot be fully optimised
DPSO		Improved availability of the resources Suitable for elasticity and scalability purposes	Fault tolerance can be supported Resources cannot be fully optimised

## **2.5 The Quest of This Research**

This section contains reviews of the key concepts of the load balancing in Cloud. As the main focus of this dissertation is on dependent and independent load balancing algorithms, to highlight the certain path of this study from reviewed dependent and independent load balancing techniques, we have selected workflow scheduling and replication methodology as the main targets of this research.

Considering these two concepts, to be aligned with the hypothesis of this research, the anticipatory approach will be embedded in these two techniques. Therefore, the next section describes the anticipatory method followed by revealing the open issues in workflow scheduling and replication strategies.

### **2.5.1 The Anticipatory Approach**

As indicated in the hypothesis of this research, the aim is to design an optimised load balancing computational model with the anticipatory behaviour blueprints. The following sections are describing the anticipatory concept in more details.

Although the words anticipation and predictions are considered to be synonyms, they have clear distinctive meaning. Prediction is illustrating a specific future event while anticipation is a future-oriented action which is applying productivity to complete the essential actions (Pezzulo et al 2008).

According to Rosen (1985) anticipatory system has the predictive functionality which makes the system aware of the instant changes. Anticipatory systems are capable of optimising the behaviour of the system with predictive functionality. These systems are aware of the future changes. Therefore with different variations, they can predict the behaviour needed to keep the systems performance stable (Kitajima, Goto & Jingde 2008).

Anticipatory behaviour is describing the future states of the systems by considering the past events. According to Butz (2008) the anticipatory behaviour not only considers the past predictions but also examines the present and future expectations of the system.

Although most of the anticipatory behaviours have same functional structures, they are different in terms of objectives and purposes.

Anticipation is structured based on predictive functionality. It is essentially important for cognitive systems as it can analyse the future actions and decisions of the system. A popular example of an anticipatory system is weather forecasting (Hayashi, Spencer & Nasuto 2013).

As illustrated above, the anticipatory approach is architected based on predictive models. The prediction approaches are the most important components of a successful anticipatory system.

Different sources exist for designing anticipatory model. The main type of this model which is mainly used in our research is the statistical prediction. This model is methodized based on statistical analysis. Applying soft computing algorithms such as neural networks, the model is able to examine the past events to anticipate the future path (Kadim 2007).

There are two types of anticipations (Miyake, Onishi & Popper 2003):

1. Effect anticipation: In this model which is a goal oriented, there should exist a goal which can result in action by a controlled anticipation.
2. Trigger anticipation: The model needs essential circumstances to trigger an anticipation that can result in an action.

In this research both of the anticipatory behaviours have been applied in the experimental designs.

## 2.5.2 Workflow Scheduling

Workflow management system is one of the main concepts that help researchers improve their work on large scale applications. In workflow management, task scheduling and load balancing were always the major concerns.

With emerging the new distributed technologies such as Cloud Computing, there is a need for a more powerful computational model that can optimise the task scheduling and load balancing in workflow structured models. Thus, variety of workflow scheduling algorithms have been designed and implemented. However, still there are many challenges that remain open in the area of workflow management (Deelman & Chervenak 2008)

Yu et al (2005) presented a comprehensive workflow management survey which highlights different approaches for building a workflow management system.

Bahsi et al (2007) presented a conditional workflow management that analyses the workflow's structure with "while, if and switch". They were emphasising on the conditional structure that controls the data flow and resource management in workflow applications.

Yu et al (2008) proposed heuristic workflow scheduling algorithm with the aim of optimising the quality of service (QoS) in an intensive workflow application.

The heuristic algorithm that was highlighted in this research could optimise the load balancing based on the deadlines and budget.

Kwok & Ahmad (1999) also recommended a static workflow scheduling technique for scheduling the directed acyclic graphs which are functioning based on the message passing features.

In workflow applications, scheduling could happen either on tasks level or jobs level. In task level, resources will be allocated based on the dependencies and properties of each task (Blythe et al. 2005). However, in job level scheduling, the combination of the different tasks should be allocated on a computing resource. The overall computing power and usage requirements will help the scheduler to select the available resources.



Different research works have been conducted that analyse the performance of the workflow scheduling on tasks level and jobs level.

Feo et al (1995) have applied greedy randomise adaptive concept which applies scheduling on task level. The algorithm is functioning based on the min-min heuristic algorithm and it is aware of any environmental changes. The result of the research optimised the total workflow execution time.

Extending this work, Braun et al (2001) noticed that during the task transfer, some of the resources should wait and be in a standby status before they could receive tasks to start processing. Therefore, they have optimised the min-min algorithm by recommending a weighted min-min approach that considers the idle status of the resources. Jobs will be allocated on those resources that have the minimum weights. The weight is calculated by counting the total idle time and the completion time of the job (Yihonget et al. 2013). Their results show that the job-level approach has a better performance compared to task level approach.

Before Cloud Computing, most of the workflow applications were executed on a Grid and Clusters network. Deelman et al (2003) presented a simulation study which showed that executing the workflow applications with small data size, the storage cost is much higher than CPU cost. They have used an astronomy data intensive application which is called montage. They have concluded that for implementing the data-intensive applications, Cloud Computing is the best option.

Broberg et al (2009) introduced a new tool called MetaCDN, which uses 'storage Cloud' to execute the tasks with low costs and high performance.

Brandic et al (2008) presented a workflow scheduling algorithm which considers the resource failure during the workflow execution. Their results showed that the failure and recovery time could increase the performance and reliability of the Cloud based-systems.

## 2.5.3 Replication Strategy

Varieties of independent methodologies have been proposed by researchers to optimise the load balancing in Cloud systems. Among these techniques in this research, we are focusing on dynamic replication method which is considered as one of the efficient load balancing optimisation methods in Cloud systems.

Replication is one of the initial methods that have been applied in distributed environments. In Cloud systems, due to the geographic distance of the sites, replication methods could copy the replicas and distribute them among sites. This procedure provides users with higher access speed and more balanced and reliable system (Rajalakshmi et al. 2014).

Tang et al (2005) proposed two replication algorithms, simple bottom up and aggregated bottom up. In these methods, based on the file's access popularities, files are selected from down level to up.

On the other hand, Shorfuzzaman et al (2008) suggested an algorithm that had the same functionality like the Tang's algorithm, but it had two phases: in first step it aggregated the files' access histories. In the second step it examined the access history catalogue to find out the popularity of the files and pre-replicate them from up to bottom layer. The results of the algorithm minimised the total execution time of the tasks by 10%.

In another paper Sashi & Thanamani (2011) designed an algorithm that balances the load by minimising the number of the replications. So in this method which is called "Modified BHR", based on the access frequency of the files, it tries to pre-replicate the files in the local region. This method reduced the unnecessary replication by storing the required replicas on local sites rather than distributing them among existing sites.

Griffioen & Appleton (1995) recommended another replication algorithm which is functioning based on graph probability. When the first file is accessed the counting number of that file is increased. After accessing the second file, a connection edge between first and second file will be created. The result of this pre-replication methodology optimised the load balancing architecture by increasing the fault tolerance of the system.

In another paper by Kroegar & Darrell (1999), an algorithm called Finite Multi-order context has been proposed which was an improvement on previously explained algorithm. FMOC architecture is based on tree structure where each node represents the files that have been accessed first. The children of those nodes are depicting the files that have been accessed after their parents. Comparing this method with previous technique, FMOC divides the access files in different tree structures. Therefore, if there is no space for adding the file in the tree, the new nodes will be ignored which is the main issue of this algorithm.

Lamehamedi & Szymanski (2007), proposed a new replication algorithm which minimises the access rate by optimising the network bandwidth usage. The method works based on file's access patterns in terms of timing. Chang (2008) also proposed an algorithm which is called latest access largest weight. The method applies greater weight to the latest accessed file. By this technique, those files with greater weight will be selected first for pre-replications.

Lei et al (2008) recommended a dynamic replication strategy. In their method files are replicated according to their access numbers, size and the network condition.

In a similar algorithm, Tungnguyen et al (2010) recommended a cost saving algorithm which pre-replicates the files with less estimated costs.

Ghilavizadeh et al (2013) noted a novel dynamic real-time algorithm that pre-replicates the files based on the network performance and workload patterns. In this method, authors are emphasising on the workload types which have a great impact on selecting the replicas to optimise the load balancing.

Among the proposed replication methods, there were few methods which presented the pre-replication approach in Cloud Computing.

## **2.5.4 Research Issues**

### **2.5.4.1 Workflow Scheduling**

Table 2.5 summarises the highlighted works that have been done for improving the load balancing in workflow scheduling models. Analysing the pros and cons of the selected workflow scheduling algorithms, most of the algorithms were focusing on optimising the performance of the system while completing the workflow execution. However, in workflow structure applications, interconnected dependencies exist between available tasks. Therefore, it is essential to consider the load flow fluctuations among the existing tasks.

With the emergence of the new technology of Cloud Computing, soon most of the data-intensive and scientific applications will run on Cloud-based systems (Pattanaik, Roy & Pattanaik 2015). However, due to the high-cost associated with deployment of these algorithms, a need exists for designing a new algorithm that is aware of the existing load between dependent tasks. Therefore, in this thesis we are highlighting the magnitude and direction of the existing load between interconnects tasks to determine the optimised method of load balancing in workflow applications.

### **2.5.4.2 Replication Methodology**

Evaluating the pros and cons of the mentioned replication techniques, although the proposed methods have had a great impact on optimising the load balancing in Cloud-based systems, a more robust replication method is required. The new method will be capable of predicting the future needs of the users so it can pre-replicate the files before the requests have been submitted for them. Applying this concept, the response time would be minimised as the system is more balanced and files could be accessed locally. Table 2.6 summarises the reviewed replication methodologies and highlights their pros and cons.

Table 2.5- Summary of the reviewed workflow scheduling algorithms

Authors	Pros	Cons
Bahsi et al (2007)	Applies conditional structure that can control the data flow and resource management	Complex algorithm structure More CPU usage
Yu et al (2008)	Applies budget and deadline as QOS metrics	More bandwidth consumption
Kwok (1999)	Applies message passing functionality in directed acrylic graphs	Did not consider the interrelations of the tasks
Blythe et al (2005)	Applies message passing in task level/job level scheduling	More CPU and memory usage in job level scheduling
Feo et al 1995	Applies greedy randomize algorithm Adaptive to environmental changes Minimised total tasks execution time	High bandwidth usage Did not consider the tasks interdependency
Braun et al 2001	Applies weighted min-min Aware of the resource status	High memory usage for storing the resource status
Deelman et al 2003	Applies montage structure workflow to minimise the storage cost in Cloud Computing	Montage CPU usage is much higher than storage cost
Broberg et al 2009	Applies MetaCDN tool for managing the workflow task with minimum makespan	The tool is not practical for high volume data
Brandic et al 2008	Applies resource failure monitoring Resulted in high availability and performance	More bandwidth consumption Did not consider the tasks dependencies Only focuses on resource failure

Table 2.6- Summary of the reviewed replications methodologies

	<b>Pros</b>	<b>Cons</b>
Tang et al 2005	Applies simple bottom up and aggregation bottom up method. Makespan is minimised	More CPU usage More bandwidth consumption
Shorfuzzaman et al 2008	Pre-replicas are chosen based on access histories Total execution time is minimised by 10%	Did not consider the effective network usage More bandwidth consumption
Sashi et al 2011	Minimised the number of the replicas by pre-locating them on local servers	More memory usage
Griffioen et al 1995	Applies graph probability function along with fault tolerance capability	More numbers of created replicas Replicas were chosen only based on access histories
Kroegar et al 1999	Applies tree structures where each node represents the first files that have been accessed and the children of that node are depicting the files that have been accessed right after their parents	If there is no space for adding the file in the tree, the new nodes will be ignored which is the main issue of this algorithm
Lamehamedi et al 2007	Applies different file access patterns Minimised access rate Minimised bandwidth consumption	More memory usage is needed for storing the access patterns
Chang et al (2008)	Greater weight is applied to the latest accessed file which will be selected as a target pre-replica	More memory and CPU usage Did not consider the effective network usage
Lei et al (2008)	Select those replicas that have been accessed more along with minimum file size and more optimised network condition	More complex algorithm More CPU usage
Tungnguyen et al 2010	Pre-replicates the files with less estimated costs	More bandwidth consumptions
Ghilavizadeh et al 2013	Select the replicas based on network performance	More CPU usage

## 2.6 Summary

In this chapter, we reviewed the main concepts of Cloud Computing and the existing challenges within this newly emerged technology.

Focusing on load balancing concerns, a variety of techniques have been proposed by researchers to improve the load balancing issues in Cloud-based systems. Among these methodologies, this research tries to optimise the load balancing issue through workflow scheduling techniques and replication strategies.

We listed and explained the details of each existing algorithms to highlight the capabilities and inefficiencies of the proposed works. According to the literature review, it is clear that most of the proposed workflow scheduling algorithms were trying to optimise the makespan by only focusing on specific tasks and jobs without considering their inter-relationships characteristics.

Also, it was observed that replication strategies have been applied in Cloud-based systems to optimise the response time by only considering the currently submitted tasks.

As the volume of the data is growing rapidly, in Cloud Computing there is a need to manage the load more efficiently. Therefore in this chapter we discovered the potential load balancing methods that could optimise the performance and load balancing of the Cloud-based systems. The rest of the chapters are trying to apply the findings through case studies and experiments.

# Chapter 3

## Research Methodology

*“Software is a great combination between artistry and engineering.” Bill Gates-1955-*

This chapter details out the research methodology of the current study. It explains the methodology adopted for this review. The research topic focuses on a range of aspects within the issues of the load balancing and task scheduling in Cloud Computing. In order to explore these aspects, the proposed methodology employed heuristic analysis with mathematical modelling approaches to improve the existing load balancing challenges in Cloud Computing.

### 3.1 Research Design

The quantitative approach is known as "true science" and it applies traditional mathematical and analytical approaches to analyse the results.

In Quantitative research approaches, hypothesis will be generated to be proved as a result of the experiments (Erzberger 2001). Therefore, the approach has been selected as a research design for this study. Quantitative approach evaluates the existing challenges based on the available facts and the information gathered from the literature review.

The hypothesis should be provable by mathematical and analytical methods which are generally shaping the experiments needed during the study. Additionally quantitative research will construct the study in a manner that allows other users to repeat the experiments and generate the similar results (Wittenberg, Tashakkori & Teddlie 2000).

Research questions and hypothesis of this study have been explained in chapter 1- Introduction of this thesis. To validate the hypothesis, two experimental scenarios and a research action study have been explored.



The first experiment applies the tensor mathematical analysis. As a result, STEM-PSO algorithm has been designed to validate the defined hypothesis. The aim of the mathematical approach is to apply tensor analysis on workflow scheduling models to predict the magnitude and direction of the load changes.

The second experiment adopts replication strategy to design a smart anticipatory replication algorithm that pre-replicates the files for addressing the future needs. The experiment employs heuristic approaches to optimise the load balancing in Cloud Computing through replication strategy. The aim of the mathematical application is to apply pre-replication strategy to anticipate the future needs of the sites. The approach pre-replicates the files with high access probability and it minimises the access delays by reducing the makespan.

## **3.2 Research Method**

The research methodology that has been adopted in this thesis combines the theoretical concepts with experimental evaluation, resulted from simulations through programming.

When dealing with scalability and massive size of the real Cloud infrastructures, simulations can assist in testing the proposed approaches in a smaller scale environment.

In the first experiment, Java Runtime Environment 8.0 has been selected for implementing the STEM-PSO design. Java is a useful high-level programming language which makes simulation much easier by providing the developers with useful existing classes and libraries.

To implement the proposed replication strategy, Cloudsim has been adopted for this research. The code has been modified and provisioning policies have been added.

Cloudsim is an open source simulation tool, which initially has been developed by the University of Melbourne and is useful for implementing the Cloud-based case studies.

It consists of various components and graphical designs that makes the Cloud applications modelling much easier (Belalem, Tayeb & Zaoui 2010 ), (Chen et al. 2015). Additionally different Cloud enterprises are using Cloudsim to simulate and test their ideas before implementing them in a large scale Cloud environment.

HP is one these companies that is using Cloudsim for their Cloud research investigations (Calheiros et al. 2011).

Cloudsim is an extensible toolkit that enables Cloud users to simulate their provisioning models in a simulated Cloud environment. Figure 3.1 is depicting the high-level overview of the Cloudsim architecture.

Cloudsim is composed of three main elements: Data centres, Virtual machines (VMs) and provisioning policies. By employing the console interface, users can easily modify or add more provisioning policies. Additionally the tool is compatible with federated Cloud environment.

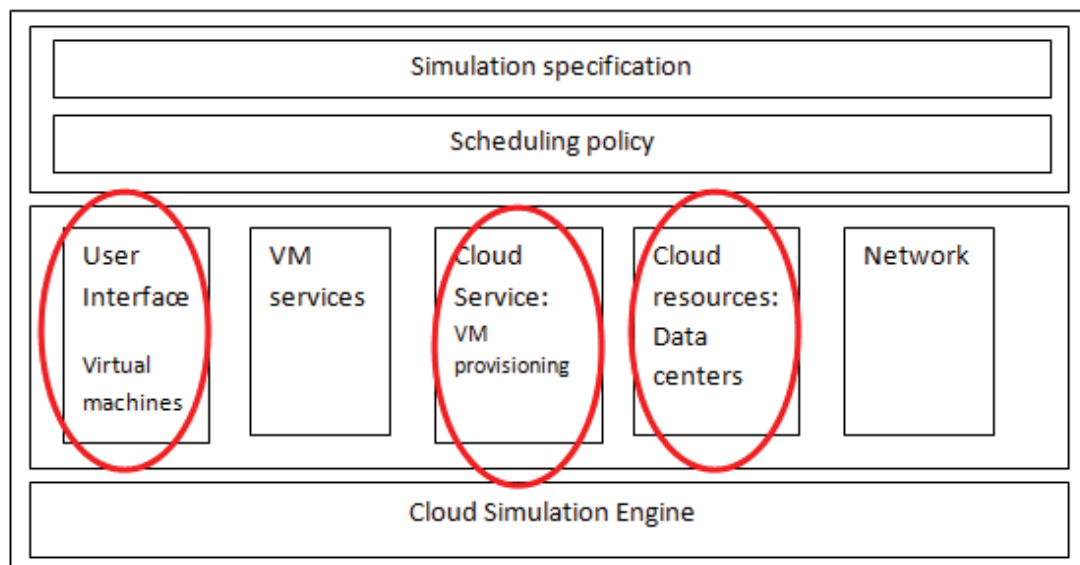


Figure 3.1 - Cloudsim architecture (Adopted from Calheiros 2011)

### **3.2.1 Research Rules**

As the aim of this research is to optimise the load balancing methods in Cloud-based systems, a certain research rules have been followed in this thesis which is described below:

1. Conduct a systematic literature review to study the previously proposed methods. This step helps to understand the general concept of the load balancing combined with investigating the existing limitations within other proposed load balancing approaches in Cloud-based systems.
2. Architect a theoretical model to solve the load balancing issues observed from step 1. This step highlights the focus of the thesis by formulating the hypothesis and developing the research questions.
3. Model and validate the theoretical model through programming and analyses the outcome results. This step designs the experiments needed to validate the theoretical model followed by implementations procedure. The expected outcomes could validate the hypothesis.

### **3.3 Modelling Approaches**

For modelling behaviour of the load balancing in Cloud infrastructure, a combination of white-box and black-box modelling has been selected (Liefsson et al. 2008):

#### **3.3.1 Black-box Modelling Approaches**

Black-box modelling approach, mathematically explains the connections between input data and output data for a specific process. In other words, black-box approach models the system's functionality by depicting the relationships between inputs and output results. Hence, in black-box modelling, there is no need to understand the theoretical details of the existing system.

Researchers applied black-box modelling in their experiments to understand the systems' behaviour. The only concern about this model is related to black-box limitation for only using the current information that have been implanted in the system. Thus, the external objects outside the derived modelling data will not be considered in black-box modelling approaches.

### **3.3.2 White-Box Modelling Approaches**

White box modelling methods are applied at the very beginning of the system designs. It considers both the external components and the internal elements that the system needs to deal with. In fact, white box modelling designs the comprehensive experimentations by applying the given hypothesis and principals of the existing system. The only concerns about white-box modelling approach could be explained as uncertainties and assumptions that are not considered in this phase. This limitation could have a negative impact on predictability of the model.

### **3.3.3 Gray-Box Modelling Approach**

Combination of the black-box modelling approach and the white-box modelling approach will result in hybrid gray-box modelling approach that improves the prediction capability of this model over black-box or white-box modelling. The reason is that, using the singular model, they only focus internal and/or external factors. Hence applying the hybrid approach, predictability will be enhanced by considering both internal and external factors. Kruchten et al (2006) suggested that the combined approaches will improve the computation path which results in enhancing the data integration methods for more accurate results.

As discussed above the aim of the gray-box modelling approach is to provide the precise optimisation model in Cloud Computing for real-time operational process. Gray-box will create a two-way relationship between white box and black box modelling approaches. White-box modelling approach will depict the actual knowledge needed to model the required experiments while black-box modelling applies this knowledge to design the internal specifications needed to model the internal elements

of the actual experiments. Figure 3.2 is depicting the hybrid approach applied to the context of this thesis

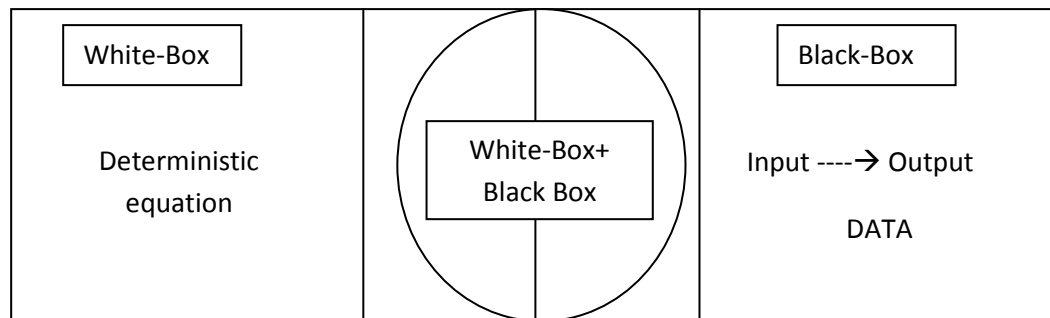


Figure 3.2 - Thesis modelling approaches (Adopted from DTU compute website)

Combination of the black-box and white-box modelling approaches assist in recognising the resources, tasks scheduling policies and protocols to ensure the robustness and reliability of the system.

As Leifsson et al. (2008) noted, Gray-box approaches could be divided to two different methods.

1. Serial gray approach
2. Parallel gray approach

In this research, we have applied serial gray-box approach which is initiating with black-box approach by modelling and pre-processing the internal elements needed for designing the system. The data then outputted toward white-box modelling.

Serial gray-box modelling approach is known as general regression model which applies to the systems that their details are not clear and needs further improvements. An example of this could be explained as obtaining the task scheduling structures and highlighting the overall system health status such as bandwidth, memory rate and ....

These elements could be captured with black-box elements and outputted toward white-box modelling approach to estimate the timespan needed to schedule the tasks. Then the estimate can be used to monitor the task scheduling and control the resources needed to optimise the system's load balancing.

## **3.4 Experimental Methodologies Overview**

### **3.4.1 Anticipatory Approach**

In this research, the experiments are designed based on anticipatory behaviour concept. The purpose of anticipatory system is to have predictive model which is aware of the system and the environmental changes. With this awareness, the system can anticipate the future changes and thus apply proper behaviour to adapt instantly to the coming changes. The details of the experiments are illustrated in section below.

### **3.4.2 Spring Tensor Model**

Advancement of ICT technology created variety of computational methods that could be applied in large-scale network systems. Within these novel computational approaches, coarse-grained methods are playing more critical role as they can deploy the experiments that can be executed over time scales which touches far beyond the network simulation capacity boundary (Jihoon et al. 2013).

Recently several coarse-grained approaches have been designed to help researchers study the complex networks where computer simulations are not feasible in that field (Wan et al. 2015).

Elastic network models (ENM) are considered as the main models of coarse-grained approaches. Among available ENM approaches, Normal mode analysis, Gaussian network model (GNM) and Anisotropic network model (ANM) are commonly used in different research projects (Cheng et al. 2012),(Vande Sande & Hameyer 2002).

Basically, GNM was adopted to show the dynamics of the proteins and then later it was officially applied to amino-acid level. GNM is able to predict the direction of the dynamics; however it can't recognise the magnitude of the elements fluctuations (Atoui, Verron & Kobi 2015).

On the other hand, reviewing the ANM capabilities, it is able to predict the

magnitude of the dynamics but it's not accurate enough in analysing the direction of the motions. Recent studies tried to combine GNM and ANM methods to use their both advantages (Sinitskiy & Voth 2013). As a result, they come up with a method called Generalise Spring Tensor model (STEM).

STEM is able to calculate the magnitude and direction of the motions; it is applied in large complex protein biomolecular structures where internal motions and relationships between large protein networks could be explored. (Clementi et al. 2000).

Generally STEM is inspired from GO-Like model, where nodes in a network are simulating the Ca atoms and they are connected to each other with a spring where there is only one degree of freedom between two connected nodes (Kalimeri, Derreumaux, & Sterpone 2015). This fact could be explained as expansion and compression motion of the spring. By providing the interactions between two nodes (e.g i and j), STEM algorithm is overcoming the deficiencies associated with GNM and ANM model (Kimura et al. 2006).

Additionally to the interactions between springs, STEM is able to consider the torsional interactions and objects bending which explain the magnitude predictions capability of the algorithm.

Besides all the advantages of the STEM algorithm the main disadvantage is related to the complexity of this method. As STEM observes the behaviour of the interconnected nodes in a whole network rather than a single section, more complexity will be added to the system (Liang & Song, 2010).

### **3.4.2.1 Mathematical Apparatus**

Generalized Spring Tensor model analyses the magnitude and direction of the load by applying the Force constant Hessian H model. Figure 3.3, is depicting a simple sub-network where 2 nodes i and j are connected and  $R_i = (x_i, y_i, z_i)$  and  $R_j = (x_j, y_j, z_j)$  are the positions of these nodes.

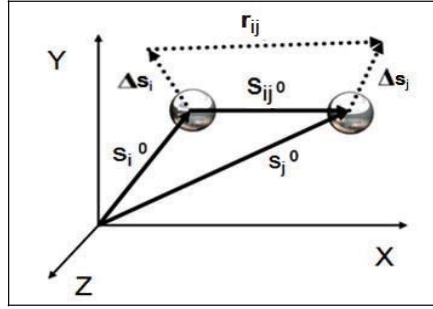


Figure 3.3- Elastic sub-network model

The distance vector between two nodes of  $i$  and  $j$  are shown as  $r_{ij}$  and equilibrium position of  $i^{th}$  and  $j^{th}$  node is depicted as  $s_i^0$  and  $s_j^0$ . Moreover,  $s_{ij}^0$  is showing the equilibrium distance between node  $i$  and  $j$  and  $s_{ij}$  represent the instantaneous range between  $i$  and  $j$ . Also  $\Delta s_i$  and  $\Delta s_j$  is showing the instantaneous fluctuation vectors. If we consider  $\gamma$  as spring constant, then the harmonic potential between two nodes can be defined as:

$$V_{ij} = \frac{\gamma}{2} (S_{ij} - s_{ij}^0)^2 \quad (3.1)$$

Then by applying the 2<sup>nd</sup> derivatives of the harmonic potential,  $V_{ij}$  is calculated as:

$$\frac{\partial^2 V_{ij}}{\partial x_i^2} = \frac{\gamma(s_{ij} - s_{ij}^0)^2}{s_{ij}^2} \quad (3.2)$$

$$\frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} = \frac{-\gamma}{s_{ij}^2} (x_j - x_i)(y_j - y_i) \quad (3.3)$$

The hessian matrix defines the force constant of the system as the second partial derivative of  $V$  potential:

$$H_{STEM} = \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ji} & H_{jj} \end{bmatrix} \quad (3.4)$$

The Hessian matrix which has  $3N \times 3N$  element can be divided to  $N \times N$  super elements where each element is  $3 \times 3$  tensor.

$$H_{STEM} = \begin{bmatrix} H_{1,1} & \cdots & H_{1,N} \\ \vdots & \ddots & \vdots \\ H_{N,1} & \cdots & H_{N,N} \end{bmatrix} \quad (3.5)$$



Each element of  $H_{ij}$  is a  $3 \times 3$  matrix that holds the tensor information. The element  $H_{ij}$  is depicting the interaction tensor between  $i$  and  $j$ .

$$H_{ij} = \begin{bmatrix} \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_i} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} \\ \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_i} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} \\ \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_i} & \frac{\partial^2 V_{ij}}{\partial x_i \partial x_j} \end{bmatrix} \quad (3.6)$$

Generally the off-diagonal super elements are calculated by:

$$H_{i,j} = \frac{1}{s_{i,j}^{p+2}} \begin{bmatrix} (X_j - X_i)(X_j - X_i) & (X_j - X_i)(Y_j - Y_i) & (X_j - X_i)(Z_j - Z_i) \\ (Y_j - Y_i)(X_j - X_i) & (Y_j - Y_i)(Y_j - Y_i) & (Y_j - Y_i)(Z_j - Z_i) \\ (Z_j - Z_i)(X_j - X_i) & (Z_j - Z_i)(Y_j - Y_i) & (Z_j - Z_i)(Z_j - Z_i) \end{bmatrix} \quad (3.7)$$

Each element of the diagonal super elements can be calculated as:

$$H_{i,j} = -\sum_{j=1, j \neq i}^N H_{i,j} \quad (3.8)$$

The  $\gamma$  is considered as spring constant where:

$$\gamma_{i,j} \propto \frac{1}{s_{i,j}^p} \quad (3.9)$$

To find out the information about the fluctuations, hessian matrix will be inversed which shows the covariance matrix of the  $3N$  multi-variant Gaussian distribution.

If  $H^+$  define the pseudo inverse of the hessian then the correlation between different residue can be defined from inverse hessian matrix:

$$\langle \Delta r_i \cdot \Delta r_j \rangle = \frac{3k_B}{\gamma} (H_{3i-2,3i-2}^+ + H_{3i-1,3i-1}^+ + H_{3i,3i}^+) \quad (3.10)$$

By examining the tensorial model; chapter 5 explores the load balancing scenario by applying STEM - PSO algorithm on workflow applications.

The quantitative analysis will define the magnitude and fluctuation of the load between non-neighbouring tasks in workflow applications and provides a global awareness in terms of task scheduling model (Chaczko et al. 2009).

### 3.4.3 Anticipatory Replication Methodology

In the second experiment of this study replication methodology have been applied to improve the load balancing in Cloud Computing. The aim is to pre-replicate the files with high access probability. Therefore whenever there is a need for accessing these files, they would be accessed locally.

The method is structured based on the following details:

**Step 1:** Each site has its own replica manager. There is also a central replication manager which monitors the sites' access and data catalogue. The central replication manager will store the files name and the numbers of the time that the file has been accessed.

**Step 2:** When a site finds that it needs a data file which is not stored locally, it asks the central replication manger to replicate the file from the available sites. Then the central replication manager will perform heuristic A\* search algorithm in the data catalogue to find out the site that has the target files. A\* is a popular heuristic algorithm, that can find the shortest path in the tree structure with minimum cost. Then between available sites the one that has the minimum bandwidth will be selected:

$$\text{communication cost} = \frac{\text{Size of the replica}}{\text{Bandwidth between servers}} \quad (3.11)$$

**Step 3:** This phase explains the replacement procedure. After replicating the files, they should be transferred to the site that was requesting the file. If the site has enough memory the file can be added easily, otherwise replacement procedure should be initiated. In replacement phase, Most Recent Used (MRU) technique has been applied which emphasises on removing the old files from the site.

So if:

$$t_{i+1} \text{ (Current time)} - t_i \text{ (last access time)} > \text{threshold old} \quad (3.12)$$

Then the old file will be replaced with a newly created replica. If still there is not enough space for the newly created replica, this replacement procedure should be

repeated until enough space created (Nader-uz-zaman et al, 2014). The details of the pre-replication methodology can be found in chapter 6 of this thesis.

### **3.5 Summary of the Chapter**

The chapter reviewed the main research methodologies that have been applied for validating the hypothesis of the thesis.

Quantitative approaches have been selected as the main research design of the experimental work. Applying the heuristic techniques along with mathematical apparatus was the main approach of justifying the hypothesis.

STEM-PSO, the first experiment, has been implemented through Java programming by employing the heuristic libraries embedded in Java Runtime Environment 8.0.

Furthermore, Cloudsim an open source Cloud simulation tool was adopted for the second experiment and extended to justify the proposed replication strategy implementation. In this simulation tool, the scheduling policies functions have been re-coded again to satisfy the anticipatory specifications of the algorithm.

As a modelling approach, gray –box modelling have been chosen which engaged both white-box and black-box modelling phases. Gray-box modelling is creating two-way relationships between white box and black box method.

White-box modelling is architecting the actual knowledge to model the external components of the experiment while black-box modelling uses this knowledge to model the internal elements of the experimental design.

# Chapter 4

## Research Action Study

*“Information theory began as a bridge from mathematics to electrical engineering and from there to computing.”- James Gleick-1954*

Availability plays an important role in Cloud-based systems. In Cloud Computing analysis of resource availability is typically performed by comparison of information abundance against resource scaling. In Cloud Computing load, balancing is a method utilised by various data centres to avoid unavailability of the network. This is achieved through the reduction in software failures, and decrease in computer hardware usage.

This chapter discusses the impact of load balancing on availability in Cloud Computing. The chapter depicts a novel load monitoring tool in the context of a case study named as Hospital Data Management (HDM). Furthermore, the case study reviles the lack of optimised load balancer algorithm when dealing with a large number of users accessing the HDM simultaneously.

As Figure 1.1 in chapter 1 (section 1.6) showed, the case study data pattern have been refined to be used for proposed load balancing optimisation methods in chapter 5 and 6 of this research work.

### 4.1 Introduction

Availability of Cloud systems is becoming of the main issues and challenges in systems that heavily rely on software for their operations. Services of Cloud systems can be interrupted for several reasons. Among these, power outages, energy/resource conservation and avoiding potential service invasion have a critical role (Hasan et al. 2012). Fundamentally availability determines the time during which the system is online and performing as required. It defines the timespan between failures and the time it takes for the system to recover and resume its normal operation following the occurrence of a failure (Chavan & Kaveri 2015).

Evaluation of availability can be performed using parameters such as current information, prediction of future usage patterns and dynamic resource scaling.

In order to reduce the possibility of outages, that could negatively impact the Cloud services, load balancing and redundant mirrored databases techniques in clusters, applied across multiple availability zones to mitigate the issue (Nadeem et al. 2008). The role of the load balancer is to accommodate transfer to a different available resource.

There are multiple benefits of applying load balancing techniques in the area of Cloud Computing. These include the reduction in costs associated with document management systems and the increase in availability of resources. The latter can reduce the negative impact on businesses by decreasing the downtime during an outage (Bonvin, Papaioannou & Aberer 2010).

In this chapter, the importance of load balancing is discussed and it is demonstrated how it can improve and maintain the availability of the Cloud systems. Further, a load monitoring tool is introduced as a case study that uses the message-oriented middleware with application of a web-service oriented model.

## **4.2 Load Balancing**

Typically Cloud vendors chose to implement automatic load balancing mechanisms into their service delivery system (Jain et al. 2013). This approach allows the numbers of resources to be adjusted with changing the demand levels. In well managed Cloud infrastructure, load balancing functions is not an option it's a must (Sarmila, Gnanambigai & Dinadayalan 2015).

The goal of the load balancers is two folds: the primary goal is to promote the availability and service provisioning of the Cloud resources, and the secondary goal is to improve the performance if needed and depending on the entities needs (Shahapure & Jayarekha 2014).

Over the years, there have been substantial improvements in load balancing techniques. Load balancers have been developing different priorities in order to increase the efficiency and performance of the distributed and multithreaded systems (Soni et al. 2014). Three basic load balancing models are used in most of the Cloud services: Server Availability model, IP Traffic Management model and Priority Queue model (Maguluri, Srikant & Lei 2012).

The Server Availability model works based on a mechanism that determines the first available server that can be utilised. Besides servers, Server Availability model can be applied to any resources (Vilutis et al. 2012). This is also the case for the other two models. Figure 4.1 shows the Server Availability model in a diagrammatic format.

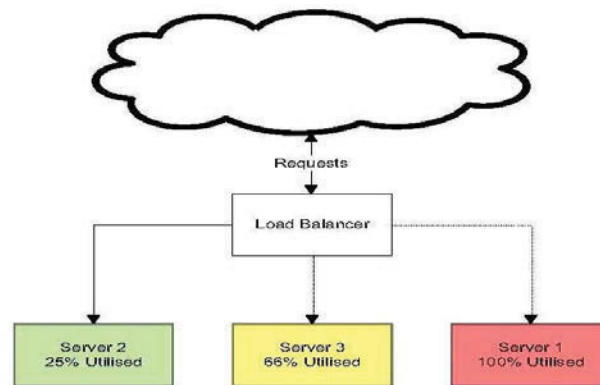


Figure 4.1-Server availability model (Chaczko et al. 2014)

In this model, availability is defined as when the server is not fully utilised and is ready/able to accept and process additional requests. There are, however, cases when a server is fully utilised and cannot accept additional requests. In such cases, the load balancer will search for the next available server (Saravanakumar & Arun 2012).

The second model, IP Address Traffic Management, considers the principal that the greater the distance (path) to the server, the longer time for the data to travel. This in turn can create issues with the transmission of the data packets. Therefore, the model works based on identifying the closest server available to accept the request

(JungYul et al. 2010). The model also ensures that the identified server can be associated with a download website using mirror sites. As a result, users are provided with a proper service at the nearby server that produces acceptable performance rather than one in a different region.

The third model, Priority Queues, works by sorting requests into categories. The categorised requests are then allocated to specific queues and a priority level is assigned to each queue (Das, Pinotti & Sarkar 1996). Services are provided and resources are allocated according to the priority of requests; i.e. the higher priority requests are processed first. The priority of resources is tied to the rank of utilisation in each network relative to other resources. The rank of utilisation is determined based on a response time, latency and utilisation of the network devices (Hoyer et al. 1995).

The load balancing models described above not only can be used individually but also can be combined to create hybrid models. For example, Figure 4.2 shows a diagrammatic representation of a hybrid model that combines the priorities and the IP traffic management features of the last two models (Ozcan & Bora 2011).

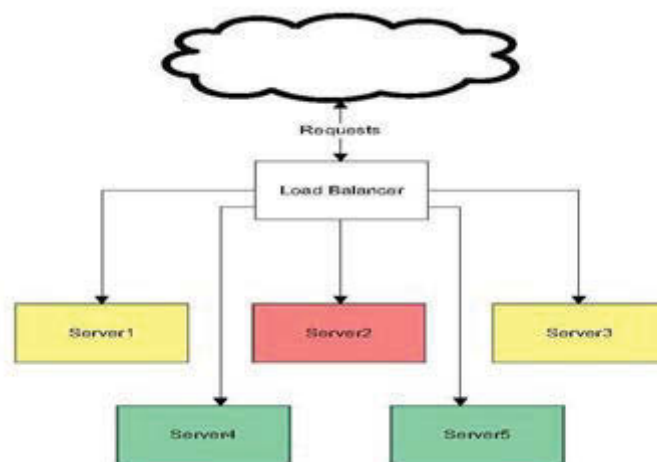


Figure 4.2- Hybrid load balancing model (Chaczko et al. 2014)

In the hybrid scenario, a request is received by the load balancer. The load balancer narrows down the server options based on the server address proximity to the location of the requestor. In this case, server 1, 2 and 3 are the closest. Because server 2 is fully utilised, the load balancer passes the request to server 1 or 3 which have not fully utilised their resources.

In this scenario applying the closer IP address management will increase the availability and performance of the system.

### **4.3 Presence**

The load balancing models described above have been developed for internal networks. In such networks, the IP address of each server and the required communication protocols are known. Use of these models in a Cloud Computing environment, however, raises some concerns. This includes identity and trust. Entities that are active in a Cloud environment have no means of knowing other entities and how to take advantage of their services (Wegscheider, Bessle & Gruber 2005). A solution to address the above issues is the use of presence. Using the presence protocol, Cloud entities can communicate easily with servers and make them aware of their existence. IETF's RFC3920 - Extensible Messaging and Presence Protocol abbreviated as XMPP (IETF 2010) have been implemented to reflect this communication concept (Xingchen et al 2012).

XMPP technology is open and used widely, mainly due to its application in online instant messaging programs (Grubitzsch & Schuster 2014). It allows for real-time communication between parties. The key, however, is the use of presence. A general overview of how XMPP works is presented below.

XMPP clients provide XMPP presence servers with the presence information. XMPP presence servers store XML streams which contain the details of presence information of clients. XMPP clients can directly access these XML streams. In other words, they can receive presence information associated with other specific XMPP clients (Ming-Ke & Yaw-Chung 2014).



Presence information, distributed in XML streams, contains a presence state, addressing information and protocols to use the advertised service, if applicable. Addressing information is typically in the form of IP address, port number and domain name (Sqalli & Sirajuddin 2005). The information related to offered services can be recorded in more than one entry for multiple services using XML markup. Presence state information was initially a Boolean on/off status. It has now developed into multiple statuses showing an entity's ability to accept incoming requests (Saint-Andre 2009). These statuses, in instant messaging, are categorised as online, busy, away, do not disturb and offline. Cloud entities implementing this protocol can advertise their services with an additional status indicator or metric. This indicator or metric will allow other entities to determine if their requests will be fulfilled promptly.

In Cloud Computing, presence information can be used to determine the availability of the Cloud entities and also for monitoring purposes. XMPP servers store presence information and process incoming and outgoing requests. Adding a load balancer to an XMPP server provides the means for prioritising incoming requests and handling them by a generic service rather than a specific entity (Hornsby & Walsh 2010).

Clustered databases are a clear example that utilise a shared-everything architecture and provide a copy of the data on each database node. In this case, applications requiring such services do not benefit from using a specific node, as the information provided is the same on different nodes (Zhen, Guo & Tracey 2007).

In addition to the previously mentioned functions, XMPP serves the common function of messaging. Resources advertise through an XMPP host in order to establish communication with it. Exchanging messages is similar to SMTP; i.e. messages are sent to the servers that deliver them to the intended recipients or the recipient's server (Hao, Binxing & Xiaochun 2006).

Messaging in a common format facilitates communication among any resource. It also establishes as how the corresponding services can be used. There could be cases of resources on another Cloud network that do not implement XMPP. In such cases, usually a gateway can be used to translate XMPP to a foreign protocol in order to

establish communications. Establishing communication with foreign networks is referred to as federation. Leveraging XMPP, in this case, will provide efficient resource monitoring with a more effective metric collection which could result in optimised load balancing (Huichao & Yongqiang 2012).

Presence information is standard only to some extent. Additional pieces of information, however, need to be included to cover deeper metrics such as CPU utilisation, memory utilisation, dynamic scaling, response time and network utilisation. These collected metrics will result in best quality of service which could also depict the availability view of the Cloud resource (Pawara et al. 2013).

## **4.4 Cloud Metrics**

One of the main characteristics in Cloud is scalability. Cloud services are able to dynamically scale based on the real-time requirements. If utilisation is excessive and beyond certain thresholds, the load balancer will usually have the option of enabling dynamic scaling for resources (Zheng et al. 2012). Depending on their requirements, utilisation thresholds can be linked to the previously mentioned metrics as well as many others. Load balancers can take advantage of the metric indicating the ability to dynamically scale and continue sending direct requests to these resources (Fiandrino et al. 2015).

If the Cloud metric indicates that a resource cannot scale, more emphasis will be placed on the other metrics. The type of the resource is important in determining what metrics is relevant to study. Observing these metrics can also reveal the costs associated with fulfilling a request (Hataba et al. 2012).

A traditional approach with regards to load balancing monitoring solutions for the Cloud has been a generic, high-level kind of approach. This, however, is not an appropriate way of dealing with such matters (Iskander et al. 2014).

These monitoring solutions lump all transactions together and consider application performance as a whole. Therefore, other factors such as background tasks that can

affect overall performance are ignored. Those with the ability to monitor transactions would only collect response times.

Transaction level metrics can determine if the load balancing is operating efficiently or more resources are needed or specific transactions should be isolated (Aceto et al. 2012). Isolating the specific transactions using metrics other than response time, allows for analysis of the transaction in terms of CPU cycles and memory. These are far better indicators to decide on load balancing.

Additionally, optimised load balancing can highlight the activity based costing. Activity based costing is a concept used in accounting. Through this concept, organisations record the costs associated with the normal business activity cycle. Subsequently, these costs can be linked to projects and business groups. Activity based costing, if combined with traced analysis, can be applied on load balancer architecture in the Cloud. This would allow for a better understanding of the resources utilisation (Chaczko et al. 2011).

Each Cloud entity could have metrics to be collected for further analysis. Monitoring the transactions performed by individual and group entities allows better utilisation of memory and improved CPU usage. It also provides an improved utilisation of the network interface connecting to the entity. This would increase availability and utilisation of the resource through the entire system (Chaczko & Aslanzadeh 2011).

The concept of trace analysis, described here, provides a complete picture of the overall design of the possible load balancer and its components. The increased metrics would be very small in size and required only for intensive transactions (Shuang et al. 2014). From a cost standpoint, however, all utilised resources should be linked to the corresponding transaction, i.e. a number of CPU cycles utilised at each point. This is because the total cost would equal to a significant amount.

Load monitoring is still a very new concept, which needs to be analysed at the transaction level within all entities. This includes resources and the connections between the resources.

All the Cloud transactions should be monitored by load balancer components. This would allow for unusual transactions to be identified, analysed and reported (Shao et al. 2015). Consequently, availability can be increased properly where needed and financial burden associated with scaling can be reduced, thus benefitting the resource owner.

## 4.5 The Case Study: HDM Load Monitoring

Considering the Cloud metrics and presence protocol described earlier, in this section a new load monitoring tool is proposed which is auditing the Cloud load in a transaction level.

The load balancer is embedded in a hospital data management Cloud system.

The case study is divided into two sections. The first part is describing the details of the novel load balancer embedded in HDM system for load monitoring. The second section, however, proves that the load balancing applied in HDM system is not optimised when dealing with large numbers of users accessing HDM simultaneously.

Consequently, the data analysis results have been refined and applied for proposed load balancing optimisation approaches in chapter 5 and chapter 6.

The Hospital Data Management (HDM) system is trying to optimise the data retrieval from multiple Clouds federated database. Figure 4.3 shows the HDM system as a separate entity relative to other databases, either in the same Cloud or different Cloud systems.

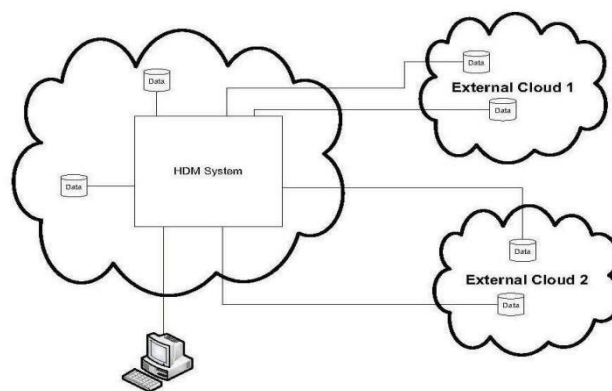


Figure 4.3- HDM System

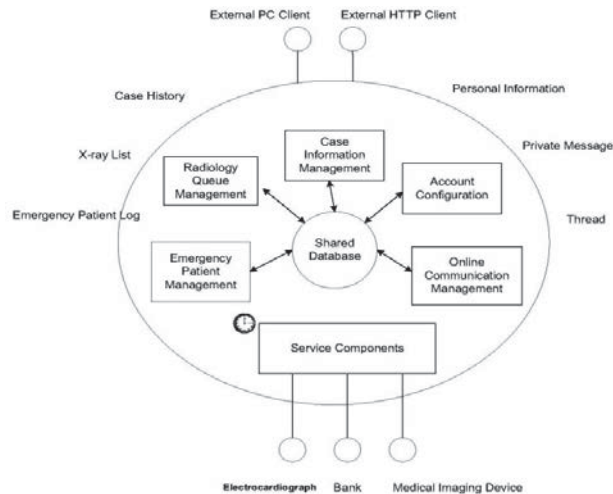


Figure 4.4- Conceptual architecture, Data Centric Model (Chaczko et al. 2011)

The HDM system will proactively scale database resources to increase availability. This is achieved by identifying usage patterns from past data.

Every doctors and nurse will have access to the HDM system. The system will be used for retrieval of patient data including various types of images such as scans, x-rays, audios and videos. Figure 4.4 shows the data-centric model of the HDM system.

The HDM system will allow users to retrieve their data without considering their location and time. This requires the system to be able to efficiently access data from different databases located on different nodes. The details of HDM system is provided in appendix, chapter 9 of this thesis.

Figure 4.5 is depicting the high-level design of the HDM database. The Resource manager class is the main component which is monitoring the load of the system. The core class is the entry class that starts the system and initialises the resource manager activities.

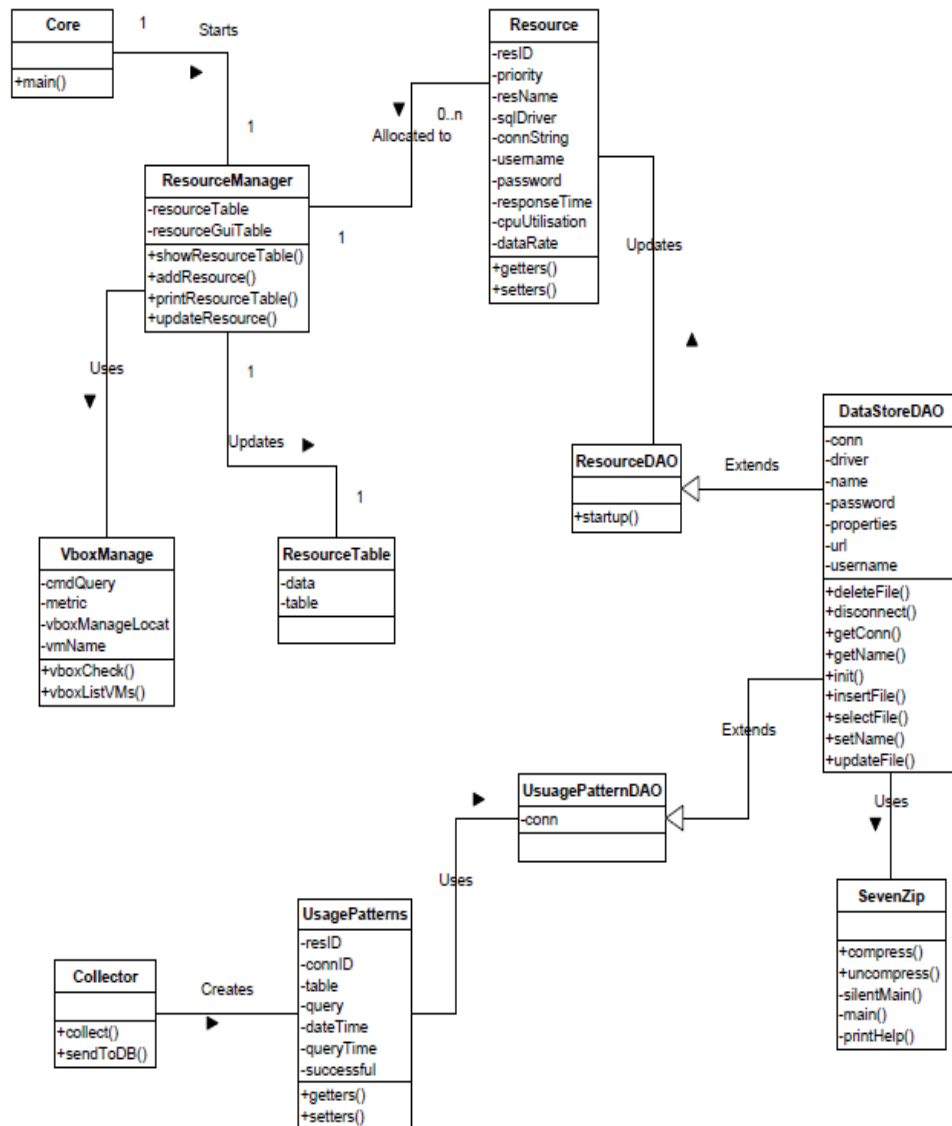


Figure 4.5- High-level HDM database design

## 4.5.1 Load Monitoring Tool

This section is highlighting a new load monitoring tool embedded in HDM Cloud system.

In HDM system by applying the presence protocol, the resource management component, is acting as a load balancer to allocate more resources where high availability of the resources matter (Figure 4.6).

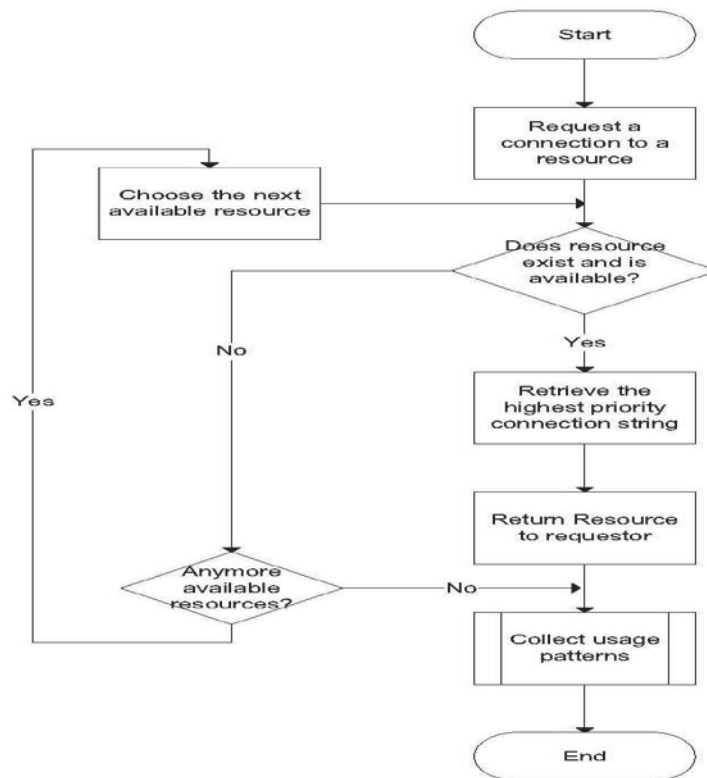


Figure 4.6- HDM load balancing model (Chaczko et al. 2011)

Resource manager updates the status of all the resources. Figure 4.7 shows the snapshot of a resource manager while monitoring the vital status of all the resources. The load-balancing package selects the highest resource according to the resource priority.

Resources are continuously monitored for availability and performance. Based on these two factors, the resource table is updated. This is achieved by calculating the new priorities with significant changes in their availability and performance. High performing resources are favoured and moved to a higher priority. Meantime, a priority of 100 is assigned to unavailable resources, which means they will never be utilised.

In this resource monitoring component, the incorporated IP address traffic management model will observe the closest IP address to the server requestor. IP address, availability and performance metrics could each be given a weighted importance to calculate the resources that should be allocated to the requestor.

Service Id	Service Type	Host	Port	Health	Response Time	Utilization %	Data rate mbps
65	MySQL	testServer4	3306	Green	10	0.23	5
23	IIS	testServer4	3066	Green	4	0.12	0.021
34	PostgreSQL	testServer4	5432	Red	400	0.7	1
35	Sybase ASE	testServer5	20003	Green	5	0	3
36	Sybase ASE	testServer5	20004	Green	5	0.02	3
40	MySQL	10.11.3.65	3306	Green	10	0.02	5
41	MySQL	10.11.3.65	3307	Green	10	0.1	5
42	MySQL	10.11.3.65	3308	Green	10	0.11	5
20	Sybase ASE	10.11.3.20	20003	Red	600	0.86	3
15	Sybase ASE	10.11.3.20	20004	Yellow	300	0.65	3
31	MySQL	testServer1	3306	Yellow	300	0.21	3

Figure 4.7- System health monitoring UI

Moreover, the resource management component can visualise the health status of all the monitored resources. Health is determined by considering response time, utilisation and data rate. A health colour is displayed according to traffic light indicators. Green health means a highly available resource while red health means low availability. If the status of the resource is coloured with white (not shown in this figure) it means that the resource is not available.

Utilising the load balancer functionality, allows the most available resources to be accessible by users and as a result data access time will be decreased dramatically.

## 4.5.2 HDM Load Tolerance

To test the load balancing function of the HDM system, we have applied load testing tools to generate virtual traffics on HDM system.

### 4.5.2.1 Testing Tool

Load tolerance testing process consists of two main components, load testing tool and page analyser that are described below:

Load testing tools are designed to simulate online traffic generated by users. The tool creates virtual users who are trying to access the HDM system simultaneously. Also, the tool can capture the speed of the page in terms of loading time from the server. This



feature allows the infrastructure team to understand how fast the HDM can response to users' requests. (Kamra et al. 2012).

3 main types of load tests can be conducted on HDM systems which are described as follow:

- Fixed – in this test the load remain static throughout the test.
- Ramp-up – in this test, load starts from a low level and will increase gradually till it reaches it's maximum capacity.
- Timeout – In this test, the load will start with high level till server goes too slow.

### **4.5.2.2 Page Analyser**

The aim of the page analyser is to replicate a scenario where a user is trying to open a HDM page in a browser. It will then order all the web objects by their loading time. It provides all these information in a statistics format.

Page analyser is using colour coding to represent the following components:

- Timespan for the first byte (green)
- Timespan for waiting tasks in queue (gray)
- Timespan for downloading the objects (blue)
- Timespan for connection time (yellow)
- DNS lookup time (orange)

This section illustrates all the load balancing tests that were conducted on the HDM Cloud based system. It tests the capability of the installed server for 4 different sets of users with a common scenario where:

- All virtual users are trying to access the website in 10 minutes.
- All users have access to the same content on the website.
- The tests have been conducted 10 times to ensure the reliability of the results for creating a valuable benchmark.
- 5 VMs with the same properties have been used in all the test scenarios. (Table 4.1)

For testing scenarios 10, 20, 50 and 100 virtual users are generated to create traffic on HDM servers. Each user tries to access medical contents which include: image.gif, application.Java script, image.vnd, text.html, text.css.

The next section depicts the results of the experiment followed by the tabulated analysis of the results.

Table 4.1- VM properties

<b>VM ID</b>	<b>VM Name</b>	<b>VM IMAGE SIZE</b>	<b>VM MEMORY</b>	<b>MIP</b>	<b>CPU NUMBER</b>	<b>Bandwidth</b>
<b>1</b>	XEN	1,000	256	250	2	1,000
<b>2</b>	XEN	1,000	512	250	2	1,000
<b>3</b>	XEN	1,000	256	300	2	1,000
<b>4</b>	XEN	1,000	512	250	2	1,000
<b>5</b>	XEN	1,000	512	250	2	1,000

## 4.5.2.3 Load Test-10 Virtual Users

### 4.5.2.3.1 10 Users-Load time vs Clients Active

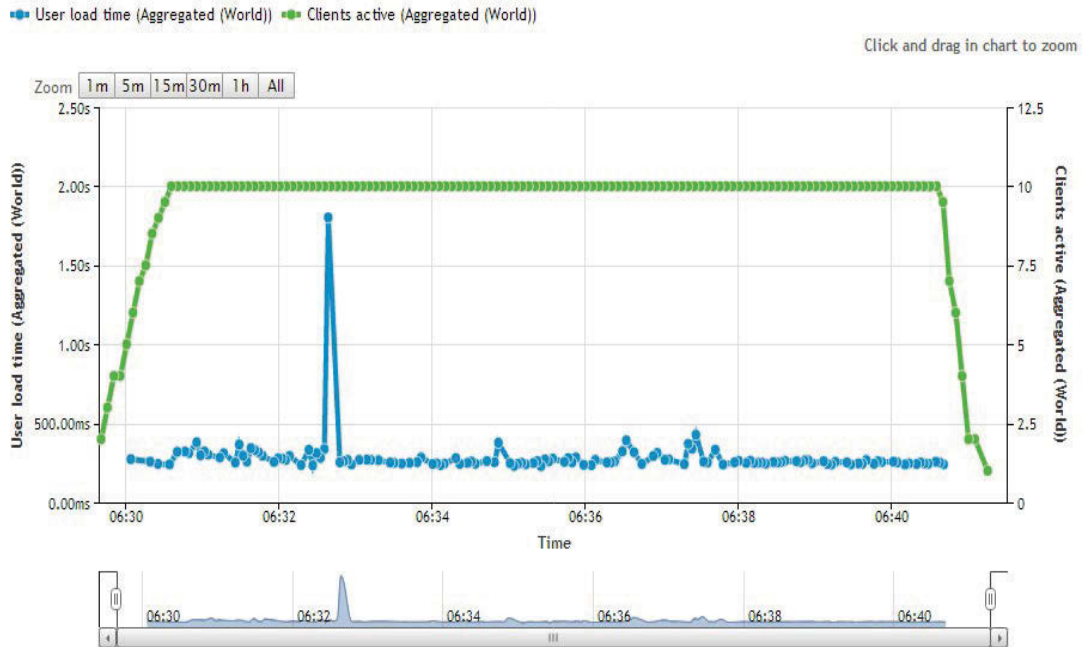


Figure 4.8- Load time vs clients active - 10 virtual users

### 4.5.2.3.2 10 Users - Number of the Active Requests vs Clients Active

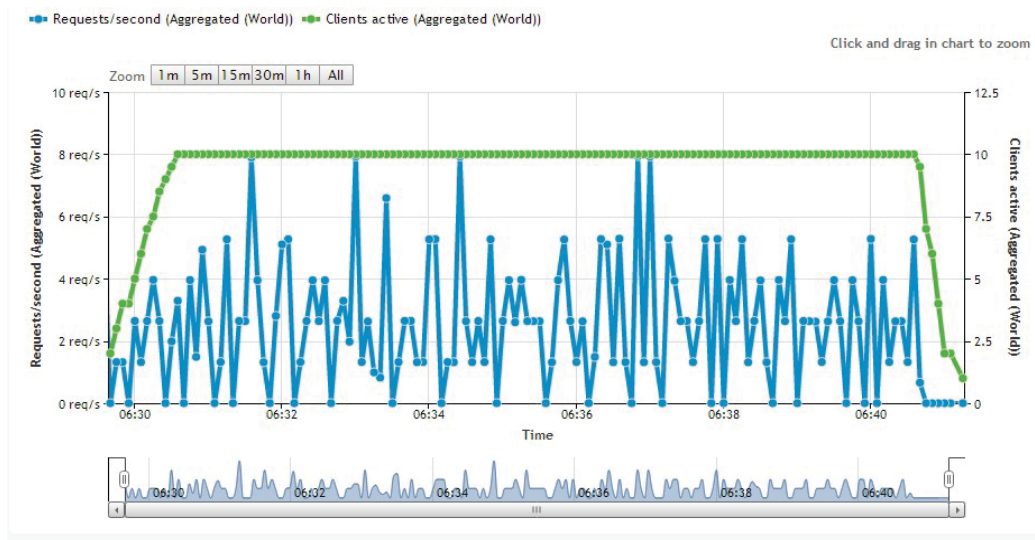


Figure 4.9- Requests per second vs clients active – 10 virtual users

### 4.5.2.3.3 10 Users - HDM Content Type Distribution

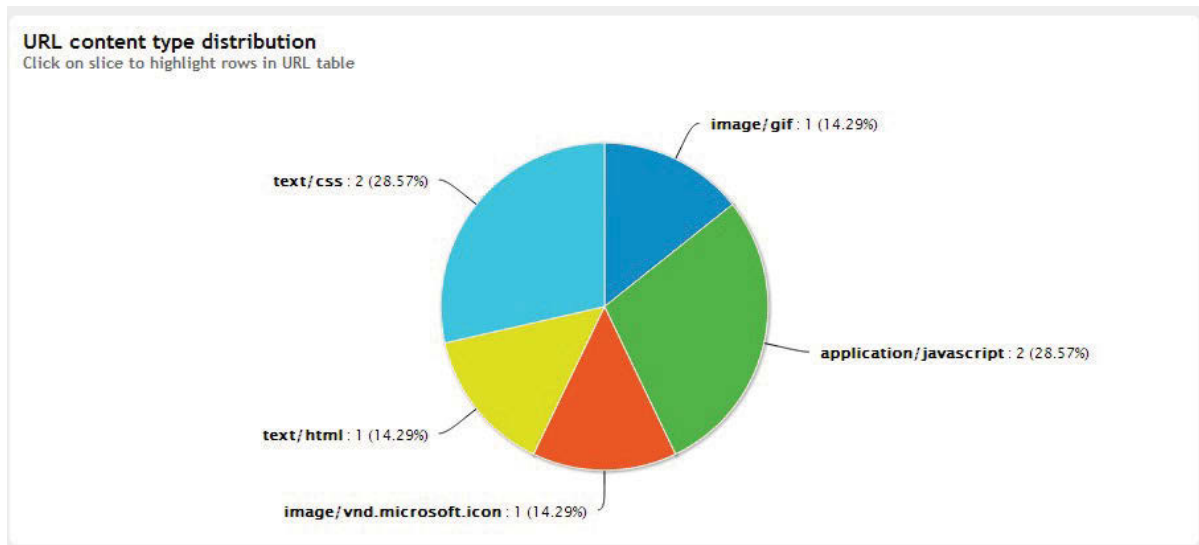


Figure 4.10- HDM content type distribution - 10 virtual users

### 4.5.2.3.4 10 Users HDM Content Load Distribution

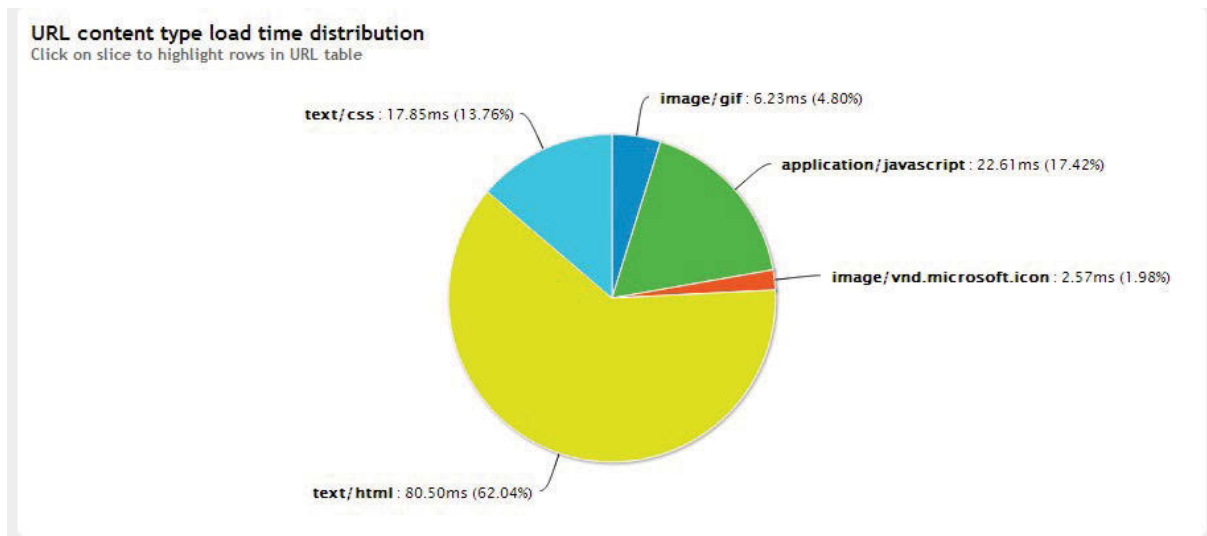


Figure 4.11-HDM content load distribution - 10 virtual users

## 4.5.2.4 Load Test - 20 Virtual Users

### 4.5.2.4.1 20 Users-Load Time Vs Clients Active

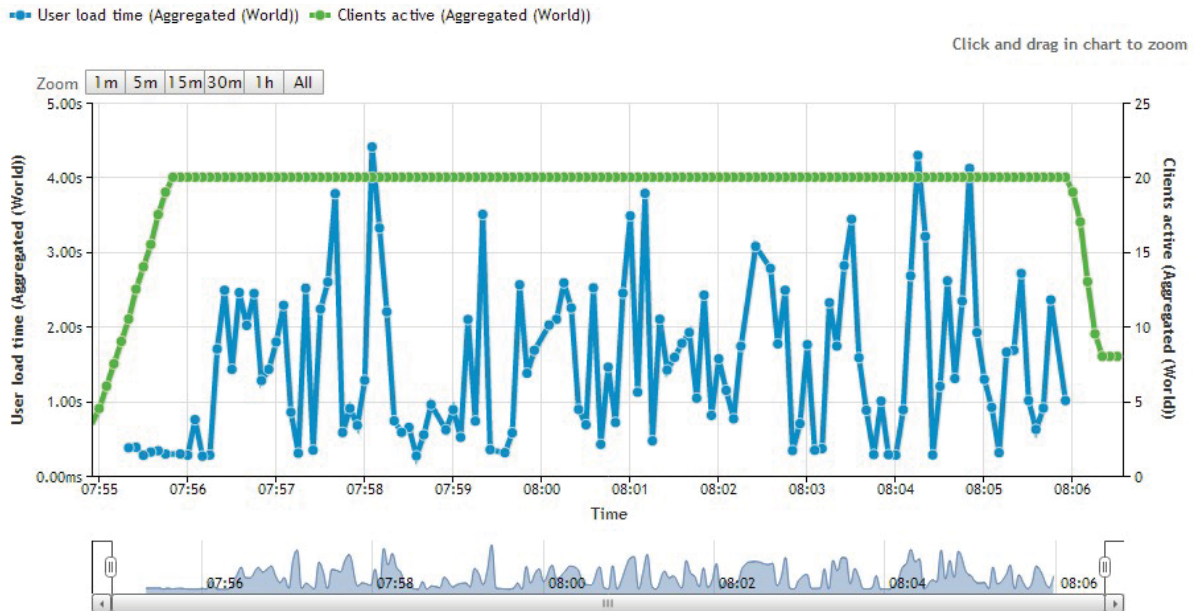


Figure 4.12- Load time vs clients active - 20 virtual users

### 4.5.2.4.2 20 Users- Number of the Active Requests Vs Clients Active

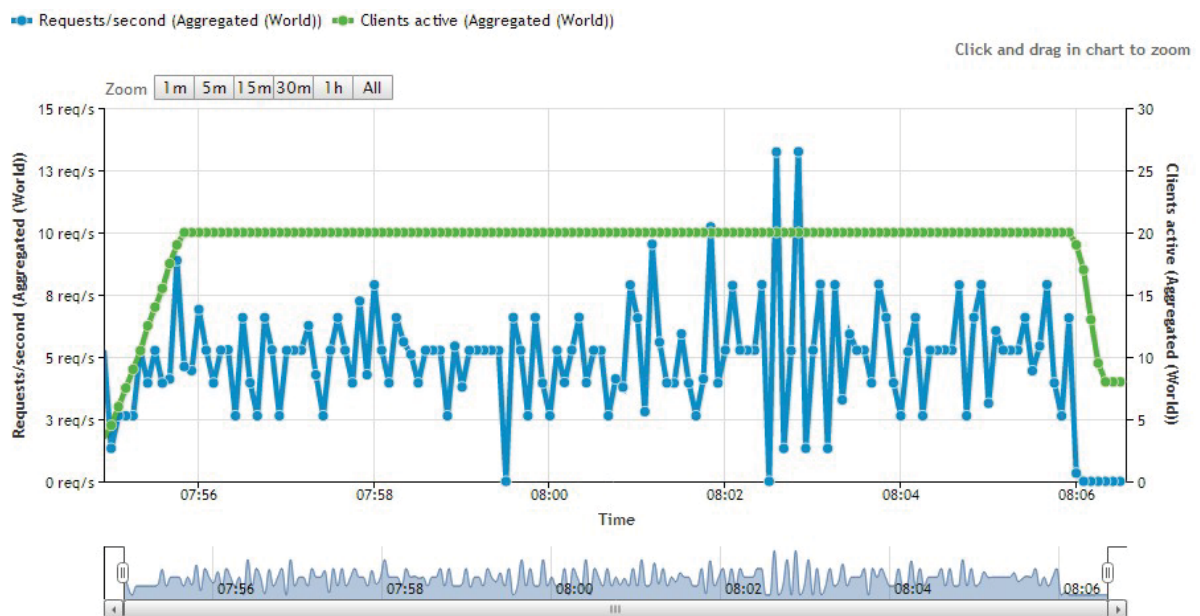


Figure 4.13- Requests per second vs clients active – 20 virtual users

### 4.5.2.4.3 20 Users - HDM Content Type Distribution

URL content type distribution  
Click on slice to highlight rows in URL table

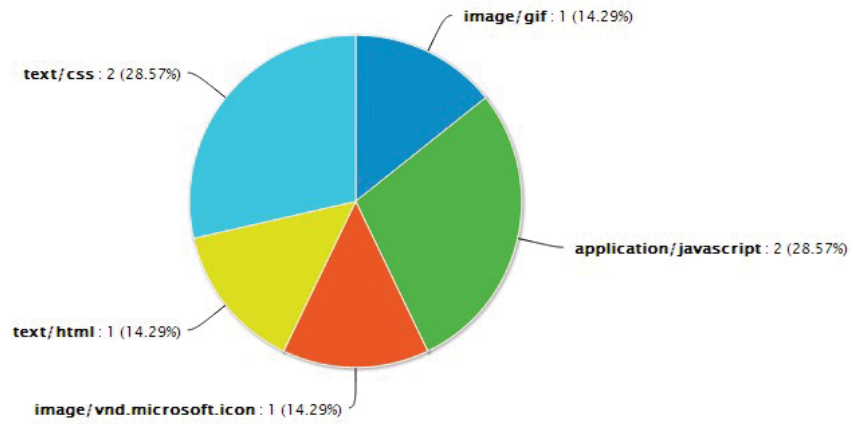


Figure 4.14- HDM content type distribution - 20 virtual users

### 4.5.2.4.4 20 Users HDM Content Load Distribution

URL content type load time distribution  
Click on slice to highlight rows in URL table

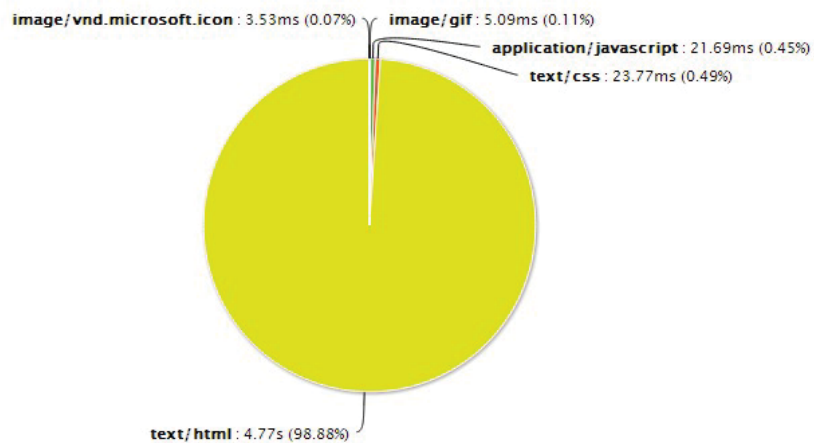


Figure 4.15- HDM content load distribution -20 virtual users

## 4.5.2.5 Load Test - 50 Virtual Users

### 4.5.2.5.1 50 Users-Load Time Vs Clients Active



Figure 4.16- Load time vs clients active - 50 virtual users

### 4.5.2.5.2 50 Users -Number of the Active Requests vs Clients Active

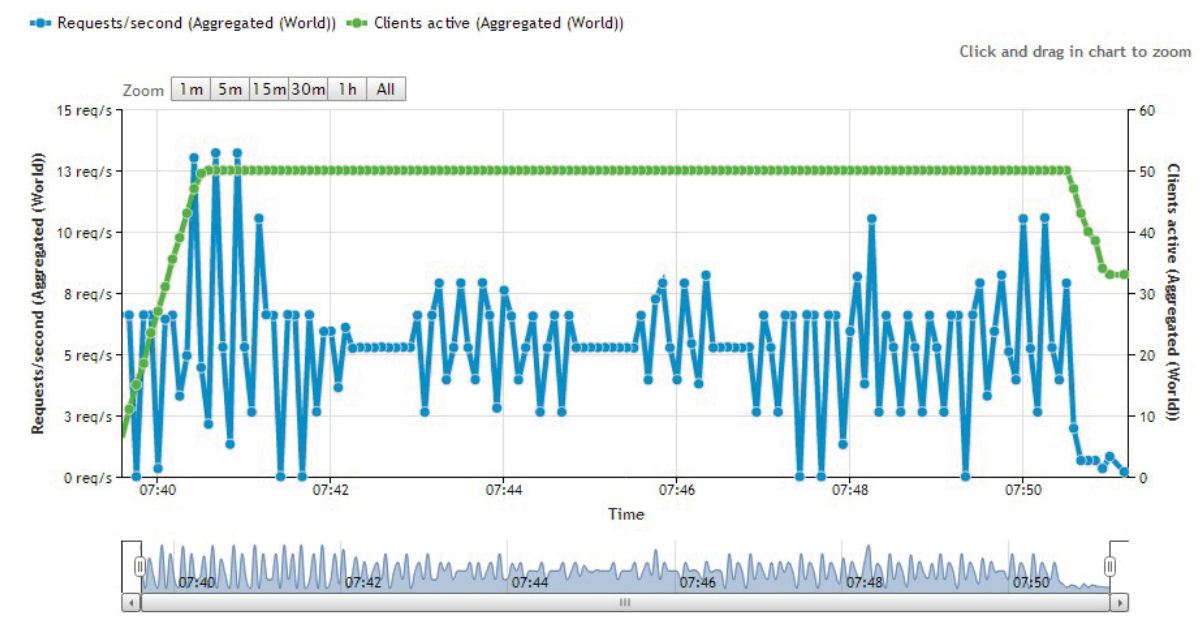


Figure 4.17- Requests per second vs clients active – 50 virtual users

### 4.5.2.5.3 50 Users- HDM Content Type Distribution

URL content type distribution  
Click on slice to highlight rows in URL table

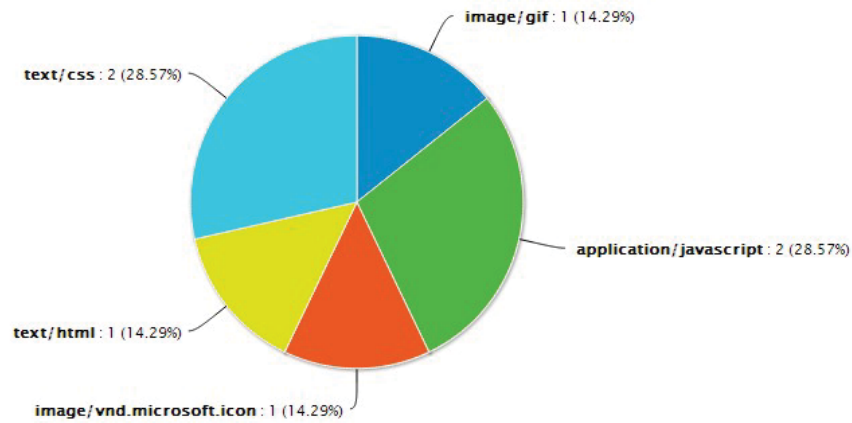


Figure 4.18-HDM content type distribution - 50 virtual users

### 4.5.2.5.4 50 Users HDM Content Load Distribution

URL content type load time distribution  
Click on slice to highlight rows in URL table

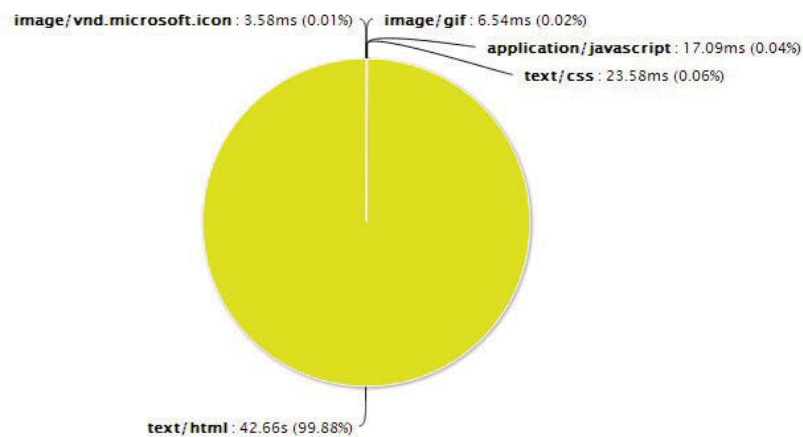


Figure 4.19- HDM content load distribution -50 virtual users



## 4.5.2.6 Load Test - 100 Virtual Users

### 4.5.2.6.1 100 Users-Load Time Vs Clients Active



Figure 4.20-Load time vs clients active - 100 virtual users

### 4.5.2.6.2 100 Users- Number of the Active Requests Vs Clients Active

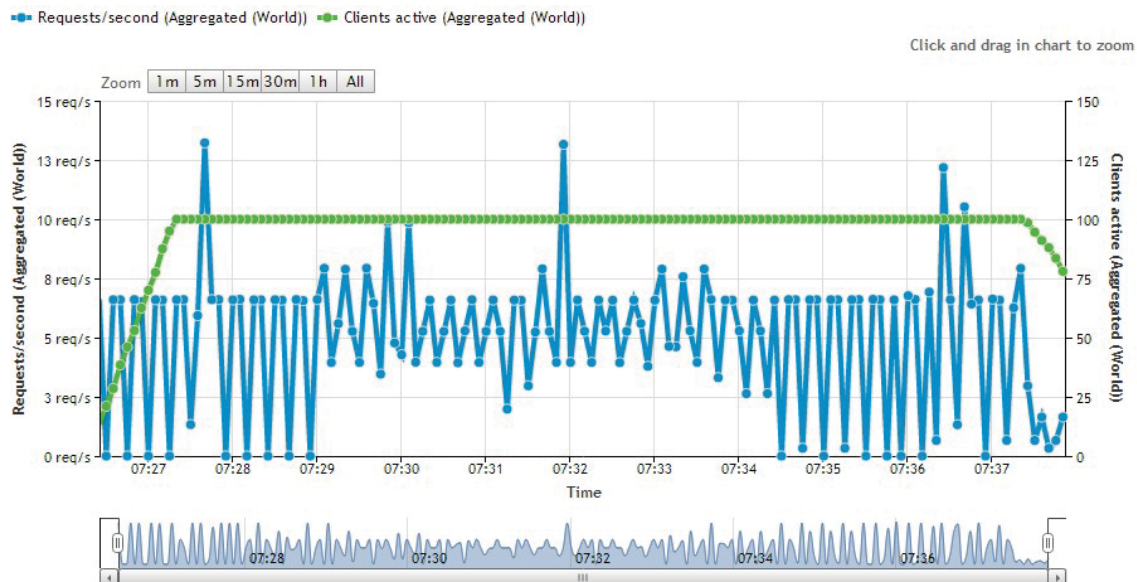


Figure 4.21- Requests per second vs clients active – 100 virtual users

### 4.5.2.6.3 100 Users - HDM Content Type Distribution

URL content type distribution

Click on slice to highlight rows in URL table

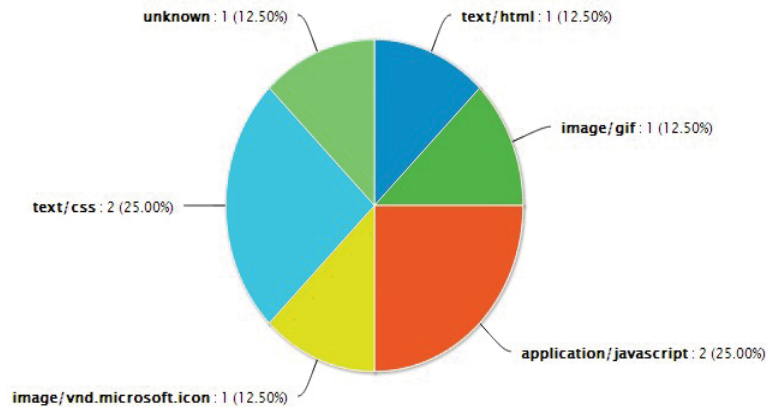


Figure 4.22- HDM content type distribution - 100 virtual users

### 4.5.2.6.4 100 Users HDM Content Load Distribution

URL content type load time distribution

Click on slice to highlight rows in URL table

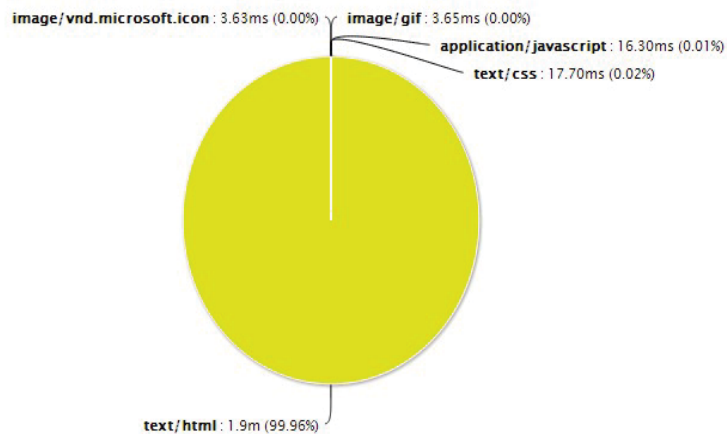


Figure 4.23 - HDM content load distribution - 100 virtual users

## 4.5.2.7 Results Analysis

### 4.5.2.7.1 Average Load Time Results

Tabulated representation of the average results for the load times is as follow.

Table 4.2- Summary of No. of clients vs average load time

No. of clients	Average Load Time for 10 tests (seconds)
10	0.315
20	4
50	44
100	109

As table 4.2 shows, it is clear that for 50 and 100 users who were trying to access the HDM simultaneously the recorded time is 44 seconds and 109 seconds respectively which is highly unappropriated for HDM performance.

Furthermore, while 20 users were trying to access HDM simultaneously, the average load time was recorded as 4 seconds which is barely acceptable.

These results may indicate that HDM did not have any efficient load balancer to distribute the load evenly.

### 4.5.2.7.2 HDM Content Type Distribution Results

The results for the HDM content type distribution (shown in table 4.3) for the load tests were observed to retain the same content type distribution percentages for the 10 tests conducted for each of the 10, 20 and 50 virtual users accessing the HDM system simultaneously:

Table 4.3-HDM content type distribution results – 10, 20, 50, 100 virtual users

Content Type	Average content type distribution results (%) with 10 users	Average content type distribution results (%) with 20 users	Average content type distribution results (%) with 50 users	Average content type distribution results (%) with 100 users
Image/gif	14.29	14.29	14.29	12.50
Text/css	28.57	28.57	28.57	25.00
Text/html	14.29	14.29	14.29	12.50
Image/vnd.microsoft.icon	14.29	14.29	14.29	12.50
Application/JavaScript	28.57	28.57	28.57	25.00
Image/gif	14.29	14.29	14.29	12.50

- Image/gif: 14.29%
- Text/html: 14.29%
- Text/css: 28.57%
- Image/vnd.microsoft.icon: 14.29%
- Application/ JavaScript: 28.57%

These results are in fact expected to be constant as all the scenarios are to access the same system which would indicate that the same data contents should be loaded for all the tests. However, for the load test that had 100 users connected showed a discrepancy when compared to the other three tests, where an Unknown segment was indicated by the load test tool.

A possible cause for this issue could be highlighted by the server timeout event which resulted in termination of the requests and left the process incomplete.

Another possibility could be related to the lack of efficient load balancer which caused the server to get overloaded and could not retrieve the data that users requested.

### 4.5.2.7.3 HDM Content Load Distribution Results

Table 4.4-HDM average content type load time distribution results – 10, 20, 50, 100 virtual users

Content Type	Average content type distribution results (%) with 10 users	Average content type distribution results (%) with 20 users	Average content type distribution results (%) with 50 users	Average content type distribution results (%) with 100 users
Image/gif	4.80	0.11	0.02	0
Text/css	13.76	0.49	0.06	0.02
Text/html	62.04	98.88	99.88	99.96
Image/vnd.microsoft.icon	1.98	0.07	0.01	0
Application/ JavaScript	17.42	0.45	0.04	0

As the results show (table 4.4), the average content type distribution for all the users are different from each other. Among the results, the test with 10 users was unacceptable but the load time was tolerable based on the available content.

However, for the subsequent content, it is clear that the distribution was only dependent on Text/html processing where it didn't allow other contents to get loaded.

As the numbers of the users grow, the processing time for Text/html loading also increases. This is a clear proof that there is a lack of efficient load balancer that can distribute the load evenly. Also, table 4.8 indicates that the average user load times conducted in the tests were directly affected by the html load time. Thus, if this html load time issue is solved, it is expected that the average user load time also decreases.

Table 4.5- Average user load time vs average html load time

No. of clients	Average load time for text/html
10	0.08
20	3
50	42
100	108
500	380

## 4.6 Overall Analysis and Proposed Solution

The results and analysis presented in the previous section depicts the need for optimising the load balancer functionality in HDM Cloud system.

Adding more virtual users to the system the result of the average load time was linear. Therefore, if more users were added on the system, the average wait time would increase forcefully.

To address this need, two algorithms have been proposed in chapter 5 and chapter 6 which can optimise the load balancing issues.

The following concepts have been verified in experimental chapters. The captured results prove a dramatic improvement in load balancing behaviour:

- Applying STEM-PSO algorithm for workflow scheduling in Cloud Computing, chapter 5.
- Applying anticipatory replication algorithm, for pre-replicating the files with high access probability, chapter 6.

The research action study provides the experimental section with refined data set patterns that include the image.gif, image.vnd, text.html and text.css.

In chapter 5, as workflow data set was needed, the HDM data set pattern was refined and used as the data input for Pegasus software to generate the workflow model with size of 64 kb to 1024 kb.

In the second experiment, chapter 6, for replication purposes the same pattern of the HDM set have been applied. A job in replication scenario contains images and texts which were replicated in 100 copies and size were refined according to the scenarios. Details of the experiment have been illustrated in chapter 5 and chapter 6 of this thesis.

## 4.7 Conclusion

This chapter discussed applying load balancing techniques to improve resource utilisation and availability in Cloud Computing environment.

There are different scheduling models and policies that can be embedded into load balancers infrastructure. These, however, should be according to the scenario that the load balancer will be used for.

Designing an optimised load balancer, the network structure or topology should be taken into consideration which could result in a different range of prices. Use of message oriented architecture as a middleware model was shown to improve the load balancing in distributed networks. Utilisation of messaging techniques, XMPP, allowed resources to be monitored efficiently. It also enhanced the availability of the Cloud resources.

In this chapter as a case study, HDM system was investigated. In the first part, a novel load monitoring tool was depicted. Applying the presence protocol, the load balancer is capable of capturing the health status of the resources. Therefore, it could optimise the load balancing by distributing the load evenly between available resources. As a result, minimum access time could be achieved.

In the second part, load tolerance of the system was tested through simulation. 10, 20, 50, and 100 virtual users were created to use the HDM system simultaneously. Analysing the load, the access time for these users were increased dramatically which made the resources unstable. Therefore as a solution, two load balancing optimisation methods have been proposed in chapter 5 and 6. The case study provided the required data patterns to be applied in experimental sections of this dissertation.

## **II. Contribution to Research**



# Chapter 5

## STEM-PSO Based Task Scheduling Algorithm

*“The engineering is secondary to the vision.”- Cynthia Ozick-1928*

In this chapter, a new load balancing algorithm is introduced that applies “Generalized Spring Tensor” (STEM) model and Particle Swarm Optimisation (PSO) to optimise the total execution time of tasks in the workflow applications.

The key objective of applying the PSO method is to minimise the total tasks execution time by verifying the load fluctuations of the interconnected tasks. The variance of the algorithm considers factors such as load variations (fluctuations) and optimisation of the data retrieval time.

The proposed model is validated by applying five workflow structures with different data block sizes. The results are compared with HEFT (Heterogeneous Earliest Finish Time) algorithm that is described in chapter 2, section 2.4.1.2.

The outcome of the experiment confirms the suitability of the optimisation method:

1. Makespan can be 2 times efficient compared to “HEFT”.
2. Better CPU and Memory usage can be achieved than “HEFT”.

## 5.1 Introduction

This chapter describes the design of the heuristic algorithm that uses PSO and STEM (described in chapter 3, section 3.4.2). The model optimises the load balancing method by minimising the total execution time.

In the Cloud architecture, in order to manage the allocated tasks, there is an application scheduler designed to balance the workload between available resources. By analysing the memory usage, processing duration and data access time, application schedulers maximise the resource utilisation.

As discussed in previous chapters, various algorithms have been designed for optimal load distribution between available resources. Particle Swarm Optimisation (PSO) is one of the load balancing algorithms that was introduced by (Kennedy & Eberhart 1995). The algorithm is focusing on the social behaviour of the particles in one population. Each particle obtains the best local position and the best global position in the entire population.

In the proposed method, each task considers as a single particle in a larger population. Chosen particles are controlled by their velocity, direction and magnitude. The novelty of the proposed model can be highlighted by projecting the load fluctuations between the parent tasks and the child tasks in a workflow application. This load aims to project the optimal mapping of the tasks on available resources.

The experiment presented in this chapter combines PSO model and STEM algorithm to project the global perception of the workflow application load. The analysis aims to distinguish the fluctuation of the load between dependent tasks in a workflow application. The results of the experiment confirm the applicability of the STEM-PSO model in Cloud Computing domain.

## 5.2 Load Balancing Problem Formulation

In Cloud Computing, load balancing is one of the major challenges that play an important role in defining the performance of the Cloud system. Without having an effective load balancer, some resources will be under-utilised and some of them will be over-utilised. Hence, to design a competent balanced system, main elements such as load estimation, load comparison and interaction between tasks and resources should be considered.

As mentioned earlier, the main objective of our experiment is to minimise the total workflow execution time which includes minimising the makespan element by considering the load fluctuations between dependent tasks.

Workflow applications enable users to perform their multi-step processing task. Workflow patterns have been applied in different scientific fields that mainly help users to retrieve and analyse workflow data in the shortest time.

Workflow model captures the hierarchical order of the tasks' execution. In order to formulate our optimised load balancing model, workflow application is examined as a Directed Acyclic Graph as shown in Figure 5.1 on page 103.

In the presented graph, the nodes define the tasks and the edges denote the dependencies between tasks and their neighbours.  $T_1$  is a root task. It generates input data for  $T_2$ , the child task.

In this experiment, for creating workflow data inputs we have used open source software called Pegasus. Pegasus is adapted to generate workflow schemas. The schemas have been used for testing the algorithm. Depending on the environment such as desktop, grid or Cloud, Pegasus creates different workflow applications that can be easily executed. Pegasus automatically provides the data required for tasks executions (Lee et al. 2008).

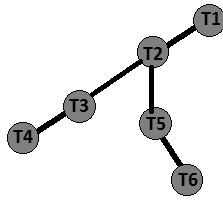


Figure 5.1- Example of workflow modelling

The model of the algorithm is formulated using the following parameters:

- The graph is presented as  $G = (V, E)$ .
- $V = \{1, 2, \dots, n\}$  shows the set of tasks where  $n$  is the number of tasks.
- $E = \{lw_{ij}\}$  denotes the load weight and the information exchange between task  $i$  and  $j$ .
- $Dl_{ij}$  presents the load weight between task  $i$  and  $j$ .
- We consider  $Bw_{ij} \ i, j = \{1, 2, 3, \dots, s\}$  as the bandwidth value between two nodes, where  $s$  is explaining the number of nodes.
- $P_j = \{1, 2, \dots, k\}$  represents  $k$  computing resources.
- $DP_{ij}$  indicates the amount of data that the task  $i$  assigning to processor  $j$ .
- $PC_m$  and  $PC_c$  are the processors memory and CPU capacity.

Based on the introduced value in above section we calculate:

$$T_{exe}(M) = \sum_{i=1}^n \sum_{t=1}^m x_{it} * DP_{it} / PC_m * PC_c \quad (5.1)$$

$T_{exe}$  denotes the task execution time where  $x_{it}$  is 1 if task  $i$  is assigned to processor  $t$  otherwise 0

- The total task transferring time can be presented as equation (5.2) where  $y_{ijkl}$  is 1 if task  $i$  is assigned to processor  $k$  ( $k \neq 1$ ) and task  $j$  is assigned to processor  $l$  for scheduling otherwise 0.

$$T_t(M) = \sum_{i=1}^n \sum_{k=1}^m \sum_{l=1}^m y_{ijkl} * Dl_{it} / Bw_{ij} \quad (5.2)$$

To validate the performance, in our proposed method, by calculating the load fluctuation as  $STEM_i$  (described in details in next section) we aim to minimise the total execution time (equation 5.3):

$$Total(M) = T_{exe}(M) + T_t(M) + (\max_{i=1}^k STEM_i) \quad (5.3)$$

Subject to  $\sum_{k=1}^m x_{ik} = 1, \forall i = 1, \dots, n$

$$\sum_{k=1}^m \sum_{l=1}^m y_{ijkl} = 1, \forall i, j = 1, 2, \dots, n, k \neq l$$

$$x_{ik}, y_{ijkl} \in \{0, 1\}, \forall i, j, k, l$$

As explained above equation (5.1) and (5.2) are representing the total task execution time and total task transforming time. Considering these two factors, equation (5.3) is trying to minimize the total task execution and transformation time by considering the fluctuation of the load calculated through STEM algorithm.

## 5.3 STEM-PSO Scheduling Algorithm

In this section, the proposed STEM-PSO is described in details. The algorithm aims to minimise the total execution time by optimising the particle swarm technique using generalized spring tensor model.

### 5.3.1 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is a heuristic algorithm proposed by Kennedy & Eberhart (1995). PSO was imitated by the social behaviour of birds and it was initially applied for balancing the weights in the neural network, but soon it was recognized as a great optimiser.

In a PSO scenario, different populations exist. Each population consists of diverse particles or individuals with magnitude and direction. Particles also have a fitness value that will be evaluated and optimised in each generation. The performance of the particles could be easily measured by their fitness value (Elshamy 2012).

The particles know their best local position (*pbest*) and the best position among the entire population (*gbest*). Considering the particle's *pbest* and *gbest*, in each movement, the particles will adjust their own current position  $x_i^k$  and velocity  $v_i^k$ . It should be mentioned that particles' positions and velocity are initiated randomly. Applying equation (5.4) and (5.5) the position and velocity of the particles will be updated respectively (Clerc 2006).

$$v_i^{k+1} = wv_i^k + c_1rand_1 * (pbest_i - x_i^k) + c_2rand_2 * (gbest_i - x_i^k) \quad (5.4)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (5.5)$$

$v_k^i$  depicts the velocity of the particle i in iteration k while  $v_i^{k+1}$  explains the velocity of particle i at iteration k+1. Position of the particle at iteration k and k+1 are shown as  $x_i^k, x_i^{k+1}$ .

$c_1, c_2$  are cognitive learning factor and  $w$  indicates the inertia weight while  $rand_1$  and  $rand_2$  are random numbers between 0 and 1.

In our case, we explain the particles as the individual tasks that should be scheduled on available resources and the dimension of the particles are representing the number of the total tasks in a workflow.

According to the number of iterations, particles will move with the light of *pbest* and *gbest*. When the iteration stops, *gbest* and fitness value are depicting the optimal scheduling strategy.

### 5.3.2 STEM Algorithm

In this work, PSO algorithm was extended, by applying Generalized Spring Tensor (STEM) model to minimise workflows' total execution time. STEM was originally proposed to project the fluctuation of the load between proteins in the human body (Chaczko et al. 2009). This algorithm has Coarse-grained mathematical structure which is composed of two main sub-models:

- Gaussian Network Model (GNM)
- Anisotropic Network Model (Uchechukwu, Li & Shen)

GNM is designed to predict the magnitude of the load and ANM mainly calculates the direction of the load.

GNM will be effective if there exists a certain dependency between tasks. This dependency can be explained by Kirchhoff Matrix (Lanoe, Le Maguer & Ney 2009) illustrated in equation (5.6) where  $r_c$  presents the cut-off distance.

$$\tau_{ij} = \begin{cases} -1 & \text{if } i \neq j \cap r_{0,ij} \leq r_c \\ 0 & \text{if } i \neq j \cap r_{0,ij} > r_c \\ \sum_{j, j \neq i}^N \tau_{ij} & \text{if } i = j \end{cases} \quad (5.6)$$

ANM, on the other hand, is applying Hookian method to simplify the measurement of the load direction. To evaluate the interactions between two nodes, ANM applies a matrix with  $N \times N$  super element, where each element is a  $3 \times 3$  tensor and  $H_{ij}$  is the interaction tensor between  $i$  and  $j$ .

$$H_{ANM} = \begin{bmatrix} H_{1,1} & \cdots & H_{1,N} \\ \vdots & \ddots & \vdots \\ H_{N,1} & \cdots & H_{N,N} \end{bmatrix} \quad (5.7)$$

ANM and GNM as two main Coarse-grained algorithms have their own advantages and disadvantages. Although they don't need energy minimization, ANM only considers the direction of the load and GNM calculates the magnitude of the fluctuation.

Thus to overcome the limitation of ANM and GNM, STEM was proposed to address the direction and magnitude of the fluctuations (Song 2012).

Workflow applications consist of collections of tasks with a variety of associated threads, commands and functionalities. Each of these tasks starts and finishes by a certain time.

The STEM algorithm recognises the workflow tasks as individual nodes; connected to their neighbours with a single spring.

Tasks in a workflow model have three main characteristics:

Tasks levels, start-time and finish-time are the main specifications of the tasks that have been applied into STEM mathematical model. Algorithm 5.1 is describing the STEM process in more details.

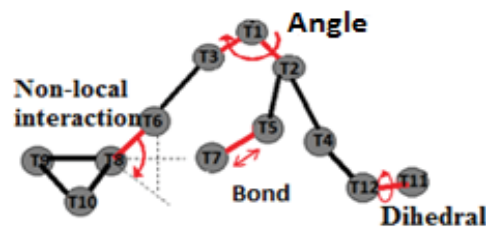


Figure 5.2- STEM workflow parameters (bond, angle, dihedral, non-local interaction)

The magnitude and direction of the workflow load are evaluated in first stage of the STEM algorithm. For this purpose, the following parameters are needed. As Figure 5.2 shows, the first derivative values for the below items are measured. Table 5.1 is illustrating the elements in more details.

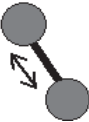


- Chain connectivity ( $r$ ), is the radial length between neighbouring nodes. In workflow model, it refers to the difference between finishing time and starting time of a task.
- Angle ( $\theta_1, \theta_2$ ), is the bonding angle between neighbouring nodes. In workflow application it is refereeing to the difference between tasks levels.
- Bond dihedral ( $\varphi$ ), is the dihedral angle between neighbouring nodes. In workflow application, it is highlighting the difference between execution lengths of the connected tasks.
- Non-local interaction ( $r_{ij}$ ), is following the radius of the non-neighbouring nodes. In workflow application, it is defining the difference between tasks levels.

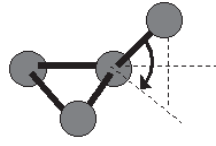


By considering the above description, the first order derivative values for bond, angle, dihedral and non-local interactions between nodes can be calculated using equation (5.8).

$$V(X, X_0) = \sum \text{Bond } V_1(r, r_0) + \sum \text{Angles } V_1(\theta, \theta_0) + \sum \text{Dihedral } V_3(\Phi, \Phi_0) + \sum \text{Non - Local } V_4(r_{ij}, r_{0,ij}). \quad (5.8)$$

Table 5.1- Interpretation of bond, angle, dihedral and non-local connections between tasks on workflow model (Aslanzadeh & Chaczko 2015).

Model	Description
	<p><i>Bond:</i> This parameter is defining the chain connectivity between tasks in a workflow model. It mainly highlights the potential connections between a task and its neighbours.</p>
	<p><i>Angle:</i> This parameter defines the angles between tasks in workflow model. The angle can be interpreted as the interval between finishing time of one task comparing with starting time of the neighbouring task.</p>
	<p><i>Dihedral:</i> The parameter describes the location of the tasks after they forced by external load. Torsion can disconnect the current connections between two nodes and it can substitute that with a new relation between non-neighbouring nodes.</p>



Non-local interaction: The parameter is defining the task's connectivity with other tasks through non-local interactions. With this model it is implied that, the force of changes between non-neighbouring tasks can be calculated.

The final step involves calculations of Hessian matrixes values. In order to calculate, the second derivative, the summation of the second derivatives of “*bond, angle, dihedral and non-local interaction*” between tasks should be considered using equation 5.9, described in Algorithm 5.1.

---

Algorithm 5.1 - STEM algorithm details

---

Step 1: Capture the level, start-time and finish-time of each task in a workflow application

Step2: Determine the Go-Like potential by calculating the first-order derivative of the chain connectivity, Angle, Bond dihedral and non-local interaction values (5.8)

The force parameters are applied to the following term:

$$v(x, x_0) = \sum_{Bond} k_r(r-r_0) + \sum_{Angles} k_\theta (\theta - \theta_0)^2 + \sum_{Dihedral} \{k_\phi^1 [1 - \cos(\varphi - \varphi_0)] + k_\phi^3 [1 - \cos^3(\varphi - \varphi_0)]\} + \sum_{non-local} \varepsilon [5 \left(\frac{r_{0,ij}^{12}}{r_{ij}}\right) -$$

$$6 \left(\frac{r_{0,ij}}{r_{ij}}\right)^{10}] \quad \text{where } \varepsilon = 0.36, K_r = 100\varepsilon, K_\theta = 20\varepsilon, K_\phi^1 = \varepsilon, K_\phi^3 = 0.5\varepsilon \quad (\text{Clementi, Nymeyer \& Onuchic 2000})$$

Step3- Obtain the Hessian Matrix values: the second order derivatives of the chain connectivity, bond bending, dihedral and non-local interactions

$$H_{ij} = \begin{bmatrix} \frac{\partial^2 V_1(r,r_0)}{\partial X_i \partial X_j} & \frac{\partial^2 V_1(r,r_0)}{\partial X_i \partial Y_j} & \frac{\partial^2 V_1(r,r_0)}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_1(r,r_0)}{\partial Y_i \partial X_j} & \frac{\partial^2 V_1(r,r_0)}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_1(r,r_0)}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_1(r,r_0)}{\partial Z_i \partial X_j} & \frac{\partial^2 V_1(r,r_0)}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_1(r,r_0)}{\partial X_i \partial Z_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 V_2(\theta,\theta_0)}{\partial X_i \partial X_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial X_i \partial Y_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_2(\theta,\theta_0)}{\partial Y_i \partial X_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_2(\theta,\theta_0)}{\partial Z_i \partial X_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_2(\theta,\theta_0)}{\partial X_i \partial Z_j} \end{bmatrix} +$$

---


$$\begin{bmatrix} \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial X_i \partial X_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial X_i \partial Y_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Y_i \partial X_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Z_i \partial X_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_3(\varphi, \varphi_0)}{\partial Z_i \partial Z_j} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial X_i \partial X_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial X_i \partial Y_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Y_i \partial X_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Y_i \partial Y_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Z_i \partial X_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Z_i \partial Y_j} & \frac{\partial^2 V_4(r_{ij}, r_{0,ij})}{\partial Z_i \partial Z_j} \end{bmatrix} \quad (5.9)$$

Step 4: the derived value determines the magnitude and direction of the load in a workflow application.

---

### 5.3.3 The Fundamentals of STEM-PSO

In our scheduling model, we have  $n$  dependent tasks that should be assigned on  $m$  processor, the position of the each task,  $\vec{X}_i = \vec{X}_1, \vec{X}_2, \dots, \vec{X}_n$  are determined by equation (5.4) and (5.5). The result of these functions should be converted to discrete values rather than continues values to determine the PCs needed to execute the task. To satisfy this goal small position values (SPV) function has been applied to convert the position values to discrete vectors  $d(X_i) = (d_1, d_2, \dots, d_n)$ .

In this model, there are  $n$  dimensions to assign  $n$  tasks on  $m$  processor with a main fitness value to minimise the total task execution time.

$$Total(M) = t_{exe}(M) + t_t(M) + (\max_{i=1}^k STEM_i) \quad (5.10)$$

The STEM-PSO algorithm is summarised as follow:

---

#### Algorithm 5.2– STEM-PSO algorithm details

---

1. Initialise the population: define the random location and velocity of tasks in the population.
  2. Convert continues position values to discrete vector using SPV method. Continues position vector of  $x_i^k = [x_1^i, x_2^i, x_3^i, \dots, x_n^i]$  should be converted to dispersed vector of  $s_i^k = [s_1^i, s, s_3^i, \dots, s_n^i]$ . In last step each vector of  $S_k^i$  should be mapped into vector  $pc_k^i = [pc_1^i, pc_2^i, \dots, pc_n^i]$  using  $p_i^k = s_i^k \text{ mod } m + 1$
  3. For each particle calculate the  $T_t$  (5.1),  $T_{exe}$  (5.2),  $STEM_i$  (Algorithm 5.1)
  4. For each particle calculate the fitness value (5.3)
  5. If the new fitness value is better than current, update the  $pbest$  value.
  6. Select the best particle as  $gbest$
  7. For all particles update the velocity and position using equation
$$v_i^{k+1} = wv_i^k + c_1rand_1 * (pbest_i - x_i^k) + c_2rand_2 * (gbest_i - x_i^k)$$

$$x_i^{k+1} = x_i^k + v_i^k$$
  8. If maximum iteration is not satisfied then
    - 8.1 Go to step3
    - Else
    - 8.2 End
-

## 5.4 Experiment Results

We have implemented our proposed scheduling algorithm by simulating a Cloud network domain through developing an application using Java programming and Jswarm package.

In this application, “Broker” is the main class which is using “bindTasksToPC” method to allocate the tasks on available resources. Jswarm, also, helps in optimising the tasks arrangement through PSO algorithm. In our proposed algorithm “bindTaskToPC” assigns tasks on resources based on Jswarm optimisation functions. A detailed description of the implementation is noted in the appendix, section 9.1.2.

The simulation environment that conducted the experiments has i5 processor, 4GB RAM and 500 HDD. We configured 4 PCs (PC1-PC4) as the main resources for task allocations. Figure 5.3 is depicting sample tasks that could be allocated on these resources. We modelled the PSO algorithm with 25 particles and 30 iterations. Moreover, Pegasus workflow management has been used to generate the input data. The size of the workflow varies in the range of 64-1024 MB.

The main evaluation component for this experiment is the total tasks execution time. To compare the performance of our proposed algorithm CPU utility and memory usage rate were also considered for analysis.

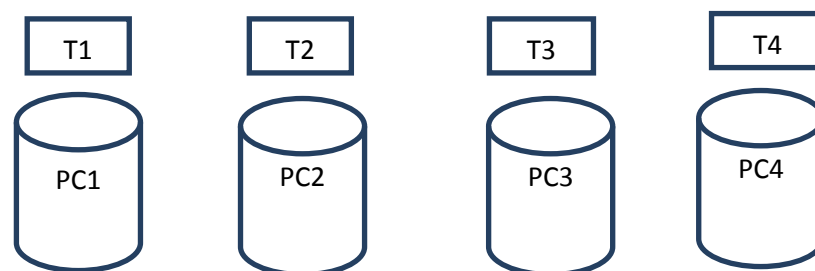


Figure 5.3- Sample tasks allocations on (PC1-PC4)

The result of the STEm-PSO optimisation algorithm is compared with Heterogeneous Earliest Finish Time (HEFT).

HEFT is a heuristic scheduling algorithm that can be applied for scheduling a set of dependent tasks on available resources. To complete the scheduling of a workflow application, HEFT considers both execution time and communication time between each connected tasks. Generally HEFT algorithm gives priority to all the workflow tasks based on their finishing time. When all the tasks are prioritised, the task with the highest priority will be scheduled on the first available resource.

### 5.4.1 Total Execution Time

Figure 5.4 presents the total time execution of the workflow. Increasing the workflow size, it appears that STEm-PSO is performing better than HEFT algorithm. The results show that the makespan calculated by STEm-PSO increases much slower than HEFT. STEm-PSO achieves at least “2 times” lower makespan for 1024 (MB) workflow size. The fact describes clearly that STEP-PSO algorithm is more efficient in terms of completing the workflow tasks (Sizes between 64 to 1024 MB) in minimum time. The STEm-PSO calculates the tasks’ execution time by considering their dependencies and load fluctuations. In workflow applications, each task can have load magnitude and direction that may vary during the execution. However, HEFT only considers the tasks’ dependencies. In HEFT algorithm, mappings of the tasks on available resources only have been calculated through local values without considering the load variations.

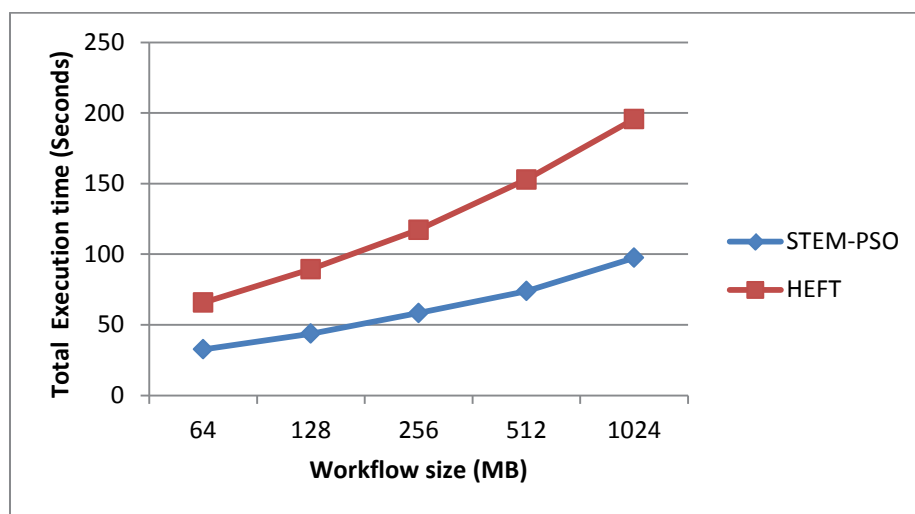


Figure 5.4- Comparison of total execution time between STEm-PSO and HEFT

## 5.4.2 CPU Load and CPU Time

We calculated the distribution of the workflow tasks onto 4 available resources (PC1-PC4) for 5 various data size (64-1024 MB). Table 5.2 is showing the details of the CPU utilisation of the workflow tasks applying the STEM-PSO algorithm.

Table 5.2 -CPU load utilisation rate using STEM-PSO

Workflow Data Size(MB)	CPU utilisation			
	PC1	PC2	PC3	PC4
64	33.23%	24.00%	25.61%	17.16%
128	50.87%	20%	16%	13.13%
256	30.76%	15%	40%	14.24%
512	36.25%	19.21%	27.67%	16.87%
1024	20%	37%	15.5%	27.5%

According to the above table, we compared the average CPU utilisation of STEM-PSO and HEFT algorithm, Figure 5.5.

Analysing the results it is clear that using STEM-PSO, CPU is more utilised than using HEFT algorithm.

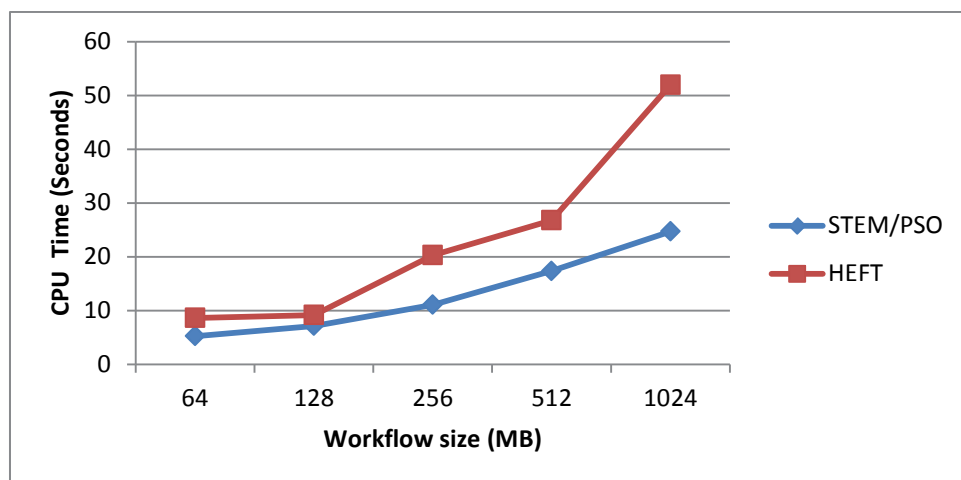


Figure 5.5- Comparison of total CPU time between STEM-PSO and HEFT

### 5.4.3 Memory Rate

In this experiment, the maximum memory rate with different values has been set to evaluate the performance of our proposed scheduling model. In this experiment different categories of memories have been considered:

1. Cache memory: This memory is functioning at the speed of CPU
2. Physical memory: This memory is operating slower than CPU
3. Virtual memory: This memory is the one that we need to check it's rates which is running on the virtual machine and it is much slower than CPU.

According to memory rate ranges from 50 to 100 (Seconds), the analysis determines that STEM-PSO algorithm performs more efficiently under different load levels. Considering graph 5.6, the analysis shows that the proposed method utilises the memory rate more efficiently than HEFT under load balancing condition.

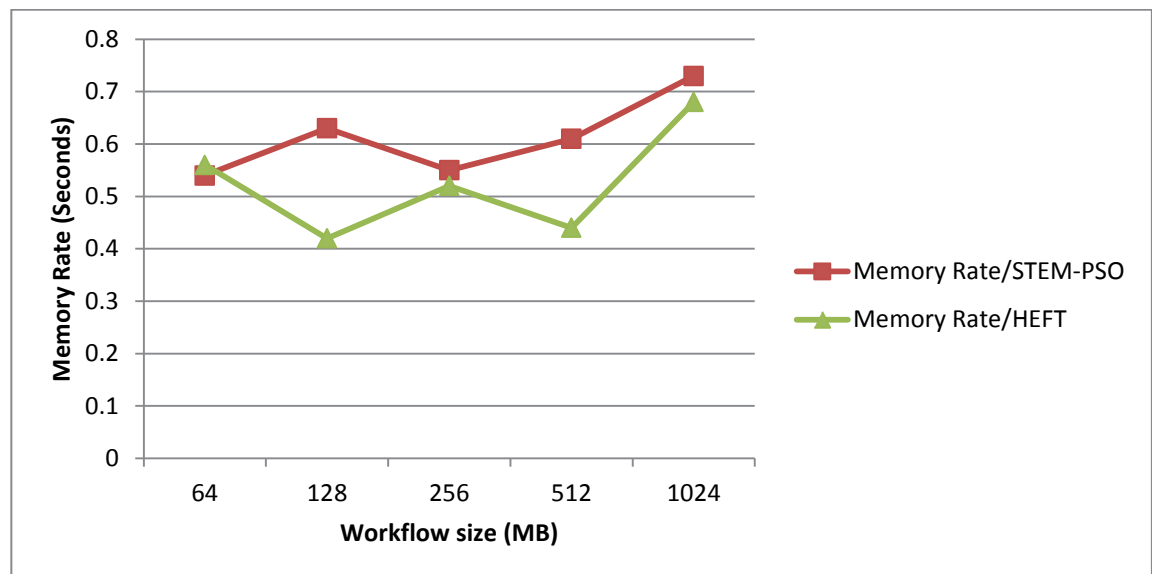


Figure 5.6-Comparison of total memory utilisation rate between STEM-PSO and HEFT



### 5.4.3.1 Workflow Soft Error Rates

To analyse the fault tolerance of the proposed algorithm we have applied soft errors on system's memory to evaluate the output results of the STEM-PSO experiment.

Soft errors usually refer to the events that could change the instructions of the program or the data value. In other words, soft errors will have an impact on data only and they won't corrupt the system's hardware. Some of the soft errors could mitigate by a cold reboot of the system. In our experiment, we have applied system-level soft error to test its effect on output results. These errors usually refer to an event when noises applied on a data bus and corrupt the data that is under process. In fact, the computer read the noise as data bits which can create errors in computing the data inputs. Therefore to apply the soft errors on memory rate we have a tool introduced by H.Belhaddad (2006) and K.Tanaka (2009) to generate soft error rates for selections of the memory blocks. 5% and 10% error rate have been applied for this scenario. 5% error, replicates the 5 events memory failure for 100-megabyte memory and 10% error interprets the 10 events memory failure for 100-megabyte memory. The results of this analysis have been depicted in table 5.3.

As the results indicate there is a dramatic difference while noise have been applied on memory blocks, therefore as future work of this research, to mitigate these soft errors, new error controllers will be applied to STEM-PSO software.

Table 5.3-Analysing the impact of soft error rates on Memory rate

<b>Workflow size(MB)</b>	<b>5% Soft error, Memory Rate/STEM-PSO</b>	<b>10% Soft error, Memory Rate/ STEM-PSO</b>
<b>64</b>	0.51	0.48
<b>128</b>	0.39	0.36
<b>256</b>	0.49	0.45
<b>512</b>	0.40	0.37
<b>1024</b>	0.63	0.58

## 5.5.4 Comparison of STEM-PSO Results

Table 5.4- Total workflow execution time applications using STEM-PSO and HEFT

<b>Workflow size(MB)</b>	<b>Total execution Time STEM-PSO (Second)</b>	<b>Total execution Time HEFT/Second</b>
<b>64</b>	32.6	65.7
<b>128</b>	43.7	89.4
<b>256</b>	58.4	117.3
<b>512</b>	73.9	152.9
<b>1024</b>	97.5	195.7

Table 5.5- CPU utilisation rate of workflow applications using STEM-PSO and HEFT

<b>Workflow size(MB)</b>	<b>CPU Time – STEM/PSO</b>	<b>CPU Time /HEFT</b>
<b>64</b>	5.23	8.62
<b>128</b>	7.14	9.17
<b>256</b>	11.08	20.28
<b>512</b>	17.34	26.78
<b>1024</b>	24.69	51.94

Table 5.6- Memory utilisation rate of workflow applications using STEM-PSO and HEFT

<b>Workflow size(MB)</b>	<b>Memory Rate/STEM- PSO</b>	<b>Memory Rate/HEFT</b>
<b>64</b>	0.56	0.54
<b>128</b>	0.42	0.63
<b>256</b>	0.52	0.55
<b>512</b>	0.44	0.61
<b>1024</b>	0.68	0.73

## 5.5.5 Experiment Analysis

Evaluating the results the following points are determined:

- **Impact of STEM-PSO on total workflow execution time**

STEM-PSO estimates the total execution time of the workflow by a mean of 52.15 across 64-1024 (MB) workflow applications with a 22.85 standard deviation shown in Figure 5.4. It can be observed that the total execution time has been improved by the average of 49% comparing with HEFT data. This highlights the significance of STEM-PSO in optimising the performance of the load balancing in Cloud systems.

- **Impact of STEM-PSO on CPU usage**

STEM-PSO depicts CPU usage of the STEM-PSO by a mean of 13.09 across 64-1024 (MB) data with a 7.96 standard deviation shown in Figure 5.5. The results show that STEM-PSO algorithm improves the CPU-usage by 43% compared to HEFT algorithm.

- **Impact of STEM-PSO on memory rate**

STEM-PSO optimised the memory rate by a mean of 0.52 across 64-1024 (MB) with a 0.10 standard deviation shown in Figure 5.6. The results show that the STEM-PSO algorithm improves the memory rate by 14% compared to HEFT algorithm.

In summation, the combination of PSO algorithm with STEM optimised the total execution time, CPU usage and memory rate of the workflow applications. The reason for these improvements can be highlighted as the capability of the STEM-PSO algorithm to calculate the fluctuation and magnitude of the load between interrelated nodes.

Applying the heuristic methodologies, STEM-PSO could enhance the load balancing by improving the resource utilisation. As a result, availability of the system could be increased greatly.

## 5.5 Conclusion

Load balancing is one of the key factors in increasing the performance of the Cloud Computing. As Cloud resources are distributed widely, it is necessary to have an optimised scheduling algorithm to distribute the load evenly between available resources and prevents the resources to be overloaded or under loaded.

Different load-balancing algorithms have been proposed for scheduling the workflow applications in Cloud Computing. Although the approaches tried to optimise the scheduling process, magnitude and direction of the load between dependent tasks were not considered. This issue can make the result less accurate.

To address this shortcoming, a heuristic scheduling algorithm has been presented in this chapter which is designed based on “general spring tensor model” and “particle swarm optimisation”.

The objective of the proposed model is to minimise the total execution time by considering the magnitude and direction of the load changes in workflow application. In fact, the algorithm is an optimisation model which minimises the task execution time and improves the resources utilisations.

To implement the proposed model, a Java based application has been developed. Comparing the results with HEFT algorithm, the analysis shows that STEM-PSO significantly reduces the executing time of the workflow tasks. Furthermore, the proposed method optimises the CPU and memory usage in compare with HEFT algorithm.

As a future work, it is possible to improve our proposed load balancing method to not only optimise the task scheduling, but also improve the energy consumption and service level agreement. Moreover, the experiments need to use a wide range of data set inputs for data analysis. The workflow applications in real Cloud network are larger than what is used in our experiment. Therefore, as a future works the experiment needs to consider the impact of the larger workflow applications with more complex data structure.

# Chapter 6

## Load balancing & Data Replication Strategy

*“Engineering is achieving function while avoiding failure”- Henry Petroski-1966*

This chapter focuses on optimising the Cloud load balancing through replication strategy. The thesis introduces a novel dynamic data replication method that is functioning based on anticipations to create the pre-replicas for future needs of the sites. The method optimises load balancing by increasing the data availability among the existing sites.

### 6.1 Introduction

Cloud Computing can process intensive applications (scientific functions) across heterogeneous environments. However, computing scientific applications need the huge amount of data which could pose more loads on the network (Chiba et al. 2010). Although different resources could be dedicated to complete the tasks, there should be a load balancing mechanism which can distribute the load proportionally between the available nodes.

Proper load balancing techniques can minimise the access latency while increasing the availability. Through load balancing, nodes will be controlled and prevented from overloading and as a result system throughput will be enhanced (Sreenivas, Pratha & Kemal 2014).

To solve the load balancing issue, replication approach is suggested as one of the load utilisation methods that can minimise the data access time (Dobber et al. 2009). Replication creates several data copies among the existing sites which dramatically impact the load balancing performance. Distributing different replicas among available sites, replication can effectively enhance data availability, system reliability and fault tolerance.

Replication is controlling the bottleneck in query processing. When a site is accessing a file remotely for several times, it would be more beneficial and more cost saving to replicate that file for site's local access (Vu, Lupu & Ooi 2010). Additionally replication can minimise the access time. As the files would be accessed locally, communication cost would be reduced. Therefore integrating load balancing and replication together could be the essential factors of any Cloud systems architecture (Yamamoto, Maruta & Oie 2005).

Considering replication techniques, it is important to recognise which files should be replicated, when replication should be created and where replicas should be stored (Ben Charrada, Ounelli & Chettaoui 2010). Generally replication could be categorised into two groups of static and dynamics:

Dealing with static replication, the replica locations are pre-defined and are unchangeable. Also, the created replicas can't be deleted unless the user deletes them manually (Loukopoulos & Ahmad 2000). The static data replication methodologies are not adaptable to any real time changes; therefore they are not suitable for processing the data-intensive applications.

On the other hand, dynamic data replications are more flexible to real-time changes and replica statuses are monitored automatically.

Furthermore, dynamic data replications are more beneficial in terms of data access cost. By dynamically replicating the required files across the data centres, data access cost is minimised while data availability is revamped (Nader-uz-zaman et al. 2014).

This chapter is proposing a new replication algorithm, called Smart Dynamic Data Replication (SDDRC). The algorithm monitors the access history catalogue to manage the existing replicas and anticipate the needs for creating pre-replicas for future needs. Analysing the algorithm's performance the response time, access latency and number of replicas were dramatically decreased.

## 6.2 Problem formulation

Replication techniques have a dramatic impact on the performance of the systems. However, it could be costly if proper replication methodology is not selected.

Therefore it is challenging to understand when replication is necessary, which files should be selected for replication, where the replicas should be stored and how the replicas should be synched with the original files (Weixiong et al. 2013).

LRU (Least Recently Used) and LFU(Least Frequently used) are popular heuristic examples of data replication methods that have been applied in different Cloud systems. Although the LRU and LFU can reduce the data access time but these replication methods are not aware of the users' future needs (Zhan-sheng et al. 2008).

In order to address the LRU and LFU limitations, there is a need for designing a smart algorithm that can anticipate the future needs of the sites and pre-replicate the data. Pre-replicating the required data, the algorithm should effectively minimise the job execution time and enhance the effective network usage so as a result load balancing would be improved.

Given the circumstances, in this research a novel algorithm has been designed that can predict the future needs of the existing sites. Based on data access catalogue, the algorithm is able to anticipate the data with high access probability that could be needed in future.

## 6.3 Proposed Methods

Pre-replication can increase the data availability and robustness of the Cloud systems and hence requested jobs can be completed with minimum execution time and high network usage output.

### 6.3.1 SDDRC Architecture Design

Figure 6.1 illustrates the high-level architecture of the proposed replication system. In this architecture, according to the optimal network transmission rate, job broker will allocate the requested jobs on the first available site.

Each site consists of data computing /storage node which is responsible for performing the job and storing all the required data. Also within each site, there is a replication manager component that is managing the files' access histories and it defines whether the requested tasks can be completed locally or remotely.

Additionally all sites are connected to the Global Replica Management System (GRMS). GRMS is creating the replicas if beneficial. GRMS consists of two main components: global data catalogue that stores the global access patterns and the pre-replication engine that anticipate the replication needs of the sites. If sites don't have the required files to complete the task, they sent the signal to GRMS to provide them with the required replicas. Then in next step, if beneficial, GRMS will send the replicas along with their adjacent files for future access of the site.



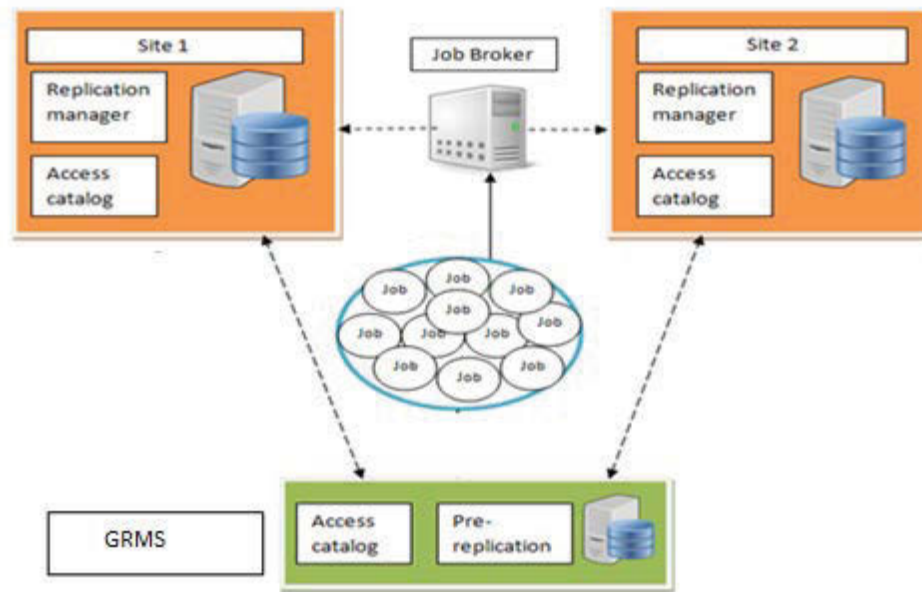


Figure 6.1– High level SDDRC system architecture

The main novelty of the proposed algorithm could be highlighted by GRMS component. As described above, GRMS is responsible for creating and replacing the replicas and pre-replicas. Figure 6.2 explains the internal architecture of GRMS. The main elements of GRMS are described below.

- Computing/Storage node: computing element is responsible for running the submitted task which stored in the queue.  
When sites request a file that is not stored locally, a copy of a requested file will be stored in the storage area.
- Pre-replica creation engine: this component is responsible for managing and creating the pre-replication data that may be requested by sites in future. In this block, prediction engine would access the data catalogue to find out the data access patterns. It would then anticipate the data with high access probability. If pre-replication is beneficial, adjacent files of the requested data also will be pre-replicated.
- Replica catalogue: when replica engine found the data that should be pre-replicated, it stores the name and physical location of that replica in the replication catalogue.

- Replica updating component: It is responsible for creating the actual replicas. By accessing the replica catalogue, this component will select the best replica with minimum communication cost.

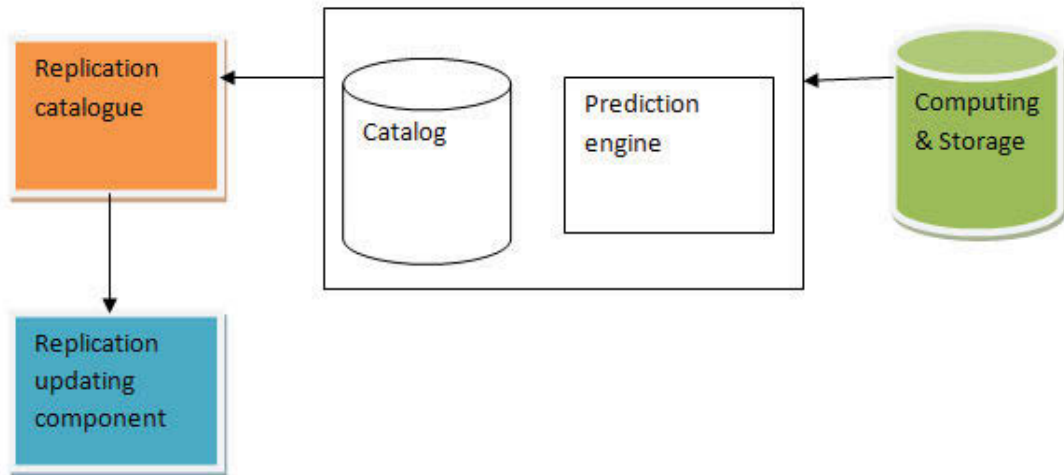


Figure 6.2- GRMS high-level architecture

### 6.3.2 SDDRC Algorithm

The previous section depicted the high-level architecture of the system. In this section, the details of the proposed algorithm will be explained. SDDRC algorithm is constructed on the basis of the following assumptions:

- For predicting the future replication needs, past sequence of access patterns should be available in the data catalogue.
- The threshold for  $t_{submitted\ file} - t_{pointer}$  is 40.
- The threshold for the maximum number of accessing the file is 10.

The detail of the algorithm is provided below:

When jobs are submitted to the system, the job broker will allocate them on available sites.

Each site consists of a computing storage manager which is responsible for analysing the required files for completing the jobs. If the computing storage manager finds that, the site has the required files then the job will be complete locally within the site.

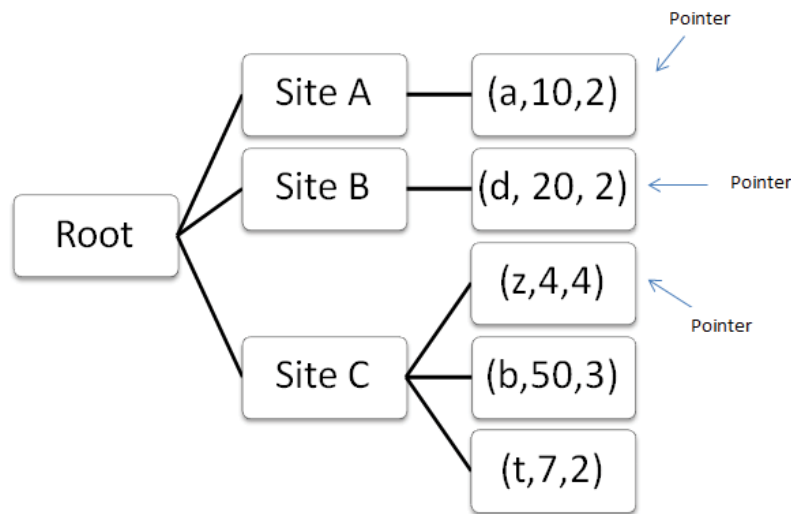


Figure 6.3- GRMS tier insertion

Otherwise, computing storage manager will send a request to GRMS and ask for the required replica.

When GRMS receives the replication request, it will start looking into the stored catalogue to check the sites access patterns.

To store this access pattern, we have applied the same tier architecture proposed in (Foster et al. 2001). Figure 6.3 is depicting the way that the tier structure stores the access pattern.

GRMS as a root of the tier will monitor the number of the sites and will store their names. This is noted as layer 2 of the tier. Layer 3 then, stores the existing files located in each site.

"File name, the latest time that the file has been requested and a number of times that the file has been accessed will be stored in each leaf of the tier. Additionally, each site has a pointer that indicates to the last file that has been accessed in the site.

When sites accessing their replica, GRMS calculate the last time that the file has been requested. If  $t_{submitted\ file} - t_{pointer} > threshold$ , it means that a new sequence line should be added under the site's name.

Otherwise if  $t_{submitted\ file} - t_{pointer} < threshold$ , it would be considered as a successive file and the child will be added to the end of the last sequence. The details of the file insertion procedure into tier structure has been explained in Appendix, section 9.1.3.1.

When computing storage component of the site notifies GRMS to find a replica, GRMS will start looking at the access catalogue to find a required file. GRMS should search in the catalogue to find the required file. Also it should consider the communication cost between the site that has the file and the site that needs the file.

Therefore to find out which site has the file, “A\*” search algorithm have been applied to find the shortest path with minimum communication cost. A\* is a popular searching algorithm for finding the best-first shortest path which satisfies evaluation function. A\* search algorithm will use heuristic approaches to avoid the path that has more expensive costs.

In A\* the evaluation function will be :

$$f(n) = g(n) + h(n) \quad (6.1)$$

$g(n)$  explains the cost for finding the file.  $h(n)$  describes the heuristically estimated cost from the source to the destination. Hence, A\* algorithm will find the target file by considering the transfer cost of the required file shown in equation 6.2:

$$communication\ cost = \frac{Size\ of\ the\ replica}{Bandwidth\ between\ servers} \quad (6.2)$$

A\* algorithm will retrieve the file with minimum communication cost. If A\* couldn't find any files that satisfy the condition it will return 0. Then GRMS will notify the site that the replication is not beneficial and job completion should be done remotely with the site that has the file.

To satisfy the anticipatory behaviour of the algorithm, GRMS will store the replica name and will look for the children of the file.

If (*number of accessing the file*) > threshold then it will add the child for pre-replication. Depending on the location of the replica and based on the business rule that is designed for this algorithm, GRMS will search 3 tiers after the replicas location and will select the file with the highest access pattern for pre-replication. If the required replica doesn't have any child it will retrieve 0 and will exit the algorithm.

Then in the last step of the algorithm, GRMS will start transmitting the replicas and its adjacent to the site that requested the files. Then replacement procedure will start.

For this replacement procedure, "Most Recently Used" algorithm (MRU) has been applied. The computing storage of the file will check if it has enough capacity for storing the received files. Otherwise, it will check the first stored replicas.

If  $t_{i+1}$  (current time) -  $t_i$  (last access time) > threshold then it will remove the old replica and insert the new replica received from GRMS. If still there wasn't enough space for storing the new replica the computing storage will continue removing the old replicas until there are enough spaces for the new replicas. The detail of the algorithm is shown in algorithm 6.1 and algorithm 6.2

## Algorithm 6.1- SDDRC Algorithm

---

1. {GRMS Store the access patterns as follow: “requested file name, requested time, and number of accessing file” }
  2. { Site request a file
  3. Search the history catalogue
    - If the requested file exist in local server
    - {then
    - retrieve the file and exit algorithm
    - else
  4. A request will be sent to GRMS.
  5. A\* search will be initiated in catalogue history to find the physical location of the file
    - 5.1 Between all the available files, calculate the communication cost and select the file with minimum value
      - communication cost = 
$$\frac{\text{Size of the file}}{\text{Available bandwidth between targeted servers}}$$
    - 5.2 For the selected replica
      - 5.2.1 check if it is beneficial to pre-replicate its adjacent files
        - {
        - check if the file has hierarchy
        - {then
        - If there exist only one child
        - { Then
        - replicate and exist
        - else
        - {
        - For 3 tiers after the replica
        - select the child with
        - maximum access number
        - }
        - }
        - }
    - 5.3 retrieve selected replica & its adjacent}
- exit
-

## Algorithm 6.2- Most Recently Used Replacement Algorithm

---

### **MRU process:**

```
{  
  1. For each new received replica  
  2. {  
  3.   If the new received replicated file.size < available storage in target server  
  4.     Insert the first replica  
  5.     Else {  
        5.1 calculate the difference between current time and last time  
        that file has been accessed  
        5.2 Select the file with minimum access number  
        5.3 Delete the file  
        5.4 go to step 3 }  
  }  
}
```

---

## 6.4 Results and Analysis

To evaluate the performance of our proposed algorithm Java programming has been used to extend the Cloudsim simulation tool. A detailed description of the Cloudsim tool is illustrated in the appendix, section 9.1.3

Cloudsim is an open source simulation package developed in Java language by the University of Melbourne. It is mainly developed to study the effectiveness of different optimisation algorithm in Cloud Computing.

In Cloudsim, we have several sites containing several virtual machines and storage elements. The broker is responsible for scheduling the requests on available resources. Each site has a replica manager that handles the automatic replica creation and deletion. Different jobs could be submitted to the broker to be assigned on available resources. The order in which the tasks should be assigned on available resources is determined by the following four main access patterns:

- Sequential: in this access pattern the files are considered as successive request and will be assigned in order.
- Random: files are accessed randomly.
- Unitary random walk: by random direction, the files are selected in a way that the successive files are exactly one element away from the previous file.
- Gaussian random walk: similar to unitary random walk with a difference that the files are selected in a Gaussian distribution.

In order to evaluate the results, SDDRC has been compared with LRU and LFU replication algorithms. These algorithms are replicating the files based on deleting the least recently used or least frequently used files.

The algorithms have been tested in four patterns: sequential, random walk, random access, random Gaussian access.

The experiment has been simulated by deploying 3 data centres, with five sites. Each site contains five VMs with 100 jobs (The same VM properties as shown in Table 4.1,



section 4.5.2.2). The minimum bandwidths between VMs are 45 Mbit/s and maximum bandwidths between sites are 10000 Mbit/s with total 10 rounds of experiments. These inputs have been hard coded in Cloudsim and will be created in runtime, Figure 6.4 and Figure 6.5.

The performance evaluation metric that have been applied in the simulations results are:

Mean Job execution time, effective network usage and the total number of replications that are described in details in next section.

```
//VM Parameters
long size = 10000; //image size (MB)
int ram = 512; //vm memory (MB)
int mips = 1000;
long bw = 1000;
int pesNumber = 1; //number of cpus
String vmm = "Xen"; //VMM name

//create VMs
CondorVM[] vm = new CondorVM[vms];

for (int i = 0; i < vms; i++) {
    double ratio = 1.0;
    vm[i] = new CondorVM(i, userId, mips * ratio, pesNumber, ram, bw, size, vmm, new CloudletSchedulerSpaceShare
    list.add(vm[i]);
}

return list;
}
```

Figure 6.4-Creating the VM data inputs in Cloudsim

```
Console [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java
<terminated>
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Starting
planner_0 is starting...
planner_0_Merger_ is starting...
planner_0_Merger__Engine_0 is starting...
planner_0_Merger__Engine_0_Scheduler_0 is starting...
Entities started.
0.0: planner_0_Merger__Engine_0_Scheduler_0: Cloud Resource List received with 1 resource(s)
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #0 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #1 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #2 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #3 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #4 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #5 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #6 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #7 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #8 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #9 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #10 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #11 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #12 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #13 in Datacenter_0
0.0: planner_0_Merger__Engine_0_Scheduler_0: Trying to Create VM #14 in Datacenter_0
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #0
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #1
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #2
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #3
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #4
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #5
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #6
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #7
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #8
4644.655: planner_0_Merger__Engine_0_Scheduler_0: Destroying VM #9
```

Figure 6.5 -Initiating the VM creating at runtime

## 6.4.1 Mean Job Execution Time

Mean job execution time is one of the important evaluation factors. Total execution of all jobs in milliseconds divided by a number of the jobs would highlight the mean job execution. To compare the performance of our algorithm, the mean job has been compared with the existing LRU and LFU algorithm. The comparison result is shown in Figure 6.6.

The simulation results show that our proposed algorithm has the lowest value in minimum job completion. As the algorithm has the functionality to anticipate the replicas, most of the files can get accessed locally. Ultimately the access time and completion time would be minimised.

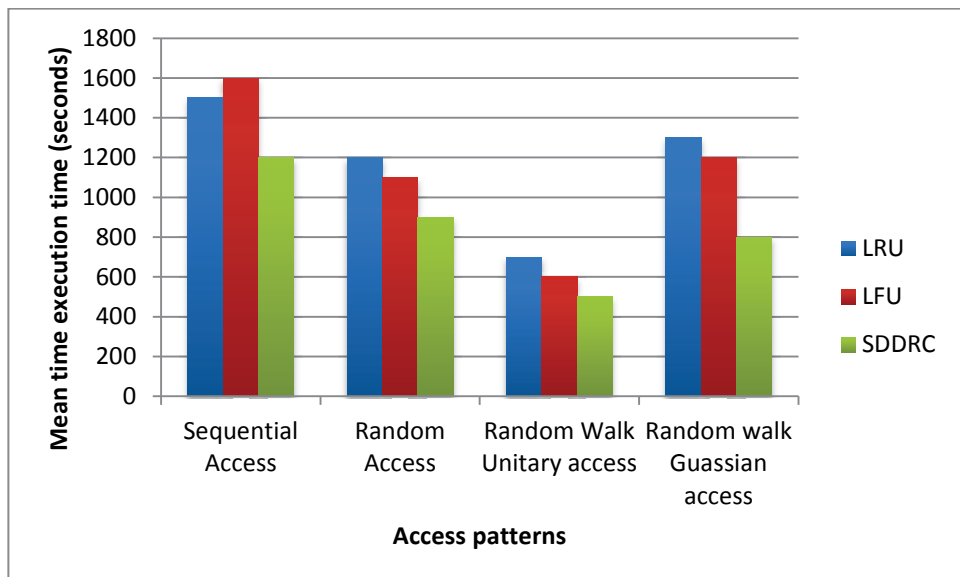


Figure 6.6- Mean job execution time

## 6.4.2 Effective Network Usage

The metric indicates the ratio of the files that were transferred to the requested site. So the low value indicates that our optimisation strategy is functioning well in our proposed algorithm. The ENU is formulated in equation 6.3:

$$\text{ENU} = (\text{N file remote access} + \text{N file replication}) / (\text{N remote file access} + \text{N local file access}) \quad (6.3)$$

The ENU of our algorithm has been compared with 3 algorithms in 4 patterns. Figure 6.7 shows that the proposed SDDRC algorithm has the lowest value in most cases which is a good indicator of the algorithm efficiency. And that's because by pre-replicating the high probable files; most of the files would be accessible locally and are available at the time of need.

In fact the higher availability of data will decrease the data replication. Therefore, as replication will not happen again, effective network usage will be decreased which has a great impact on load balancing. It should be mentioned that wrong prediction and wrong pre-replicating the files will increase the ENU and consequently it will not have any benefits for the target site rather than consuming more bandwidth.

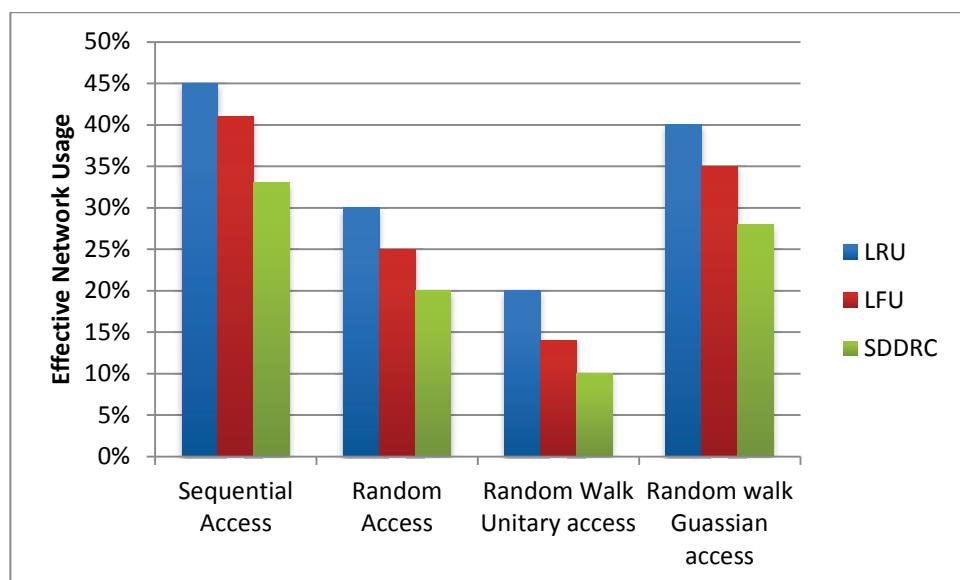


Figure 6.7- Effective network usage

### 6.4.3 Total Number of the Replicas

Greater values of the replication numbers indicate that the files were not stored locally and replication procedures were needed to make the files available.

As it is obvious in Figure 6.8 our proposed algorithm provides the lowest number of the replications comparing with LRU and LFU. It predicts the future needs of the network and estimates the files that needed to be locally accessible.

Therefore, as the files are pre-replicated before they have been actually requested, at the time of the request files would be locally accessible and no replication is needed. As a result, total replication number of the files would be decreased.

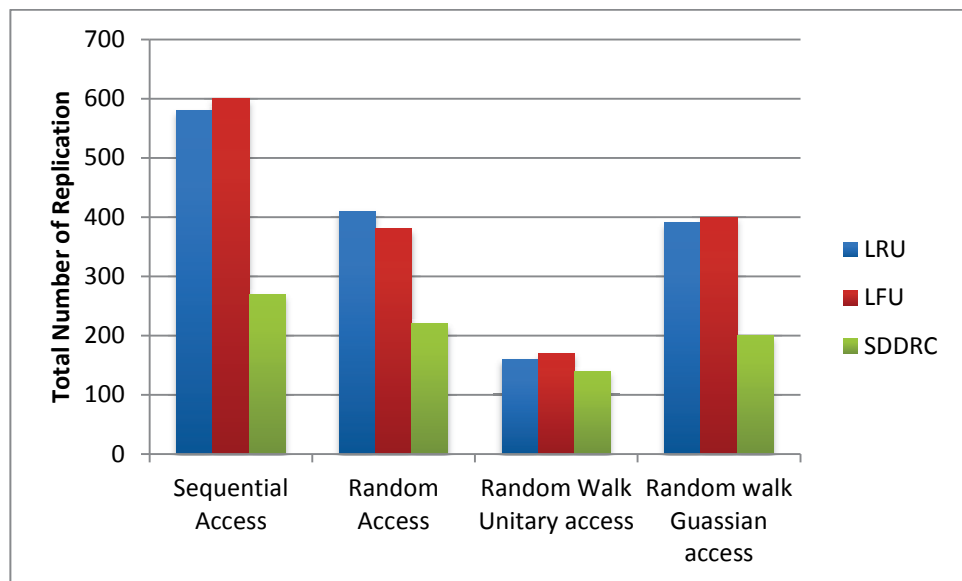


Figure 6.8 -Total number of replications

### 6.4.3.1 Applying Soft Error Rates on Memory

To consider the fault tolerance of the algorithm, as described in chapter 5 section 5.4.3.1, system-level soft error rates have been applied on input data set to analyse its impact on replication numbers. With this procedure, the memory will store the noise as the data bit that corrupts the data input. Therefore, 5% and 10% error rate are considered for analysing this scenario. Using 5% error, 5 events memory failure for 100-megabyte memory has been created and 10% error interprets the 10 events memory failure for 100-megabyte memory. The results of this analysis have been depicted in table 6.1.

As the results show, applying the noise on memory, the numbers of the replications have been increased dramatically. Therefore as future work of this research, to mitigate these soft errors, new error controllers will be coded on each site and GRMS.

Table 6.1 -Analysing the impact of soft error rates on memory rate

<b>Workflow size</b>	<b>5% Soft error, Memory Rate/STEM-PSO</b>	<b>10% Soft error, Memory Rate/ STEM-PSO</b>
<b>Sequential</b>	310	325
<b>Random</b>	298	312
<b>Unitary random walk</b>	205	215
<b>Gaussian random walk</b>	270	283

## 6.4.4. Analysis of SDDRC Results

Table 6.2 - Mean job execution time using SDDRC (seconds)

Access patterns	LRU	LFU	SDDRC
Sequential access	1500	1600	1200
Random Access	1200	1100	900
Random walk unitary access	700	600	500
Random walk Gaussian access	1300	1200	800

Table 6.3 - Effective network usage using SDDRC (throughput in percentage)

Access patterns	LRU	LFU	SDDRC
Sequential access	45%	41%	33%
Random Access	30%	25%	20%
Random walk unitary access	20%	14%	10%
Random walk Gaussian access	40%	35%	28%

Table 6.4 - Total number of replication using SDDRC

Access patterns	LRU	LFU	SDDRC
Sequential access	580	600	270
Random Access	410	380	220
Random walk unitary access	160	170	140
Random walk Gaussian access	390	400	200

## 6.4.5 Experiment Analysis

Evaluating the results the following points are determined:

- **Impact of SDDRC on mean job execution time**

SDDRC estimates the total execution time of the jobs by a mean of 850(seconds) across four different available patterns (Sequential access, random access, random walk unitary access, random walk Gaussian access) table 6.2. Comparing with LRU and LFU, it can be observed that the total execution time has been improved greatly by the average of 24% and 27% (in seconds) respectively which is significant improvement.

- **Impact of SDDRC on effective network usage**

SDDRC improved the network usage by mean of 13.09% across four different patterns table 6.3.

Comparing with LRU and LFU it can be observed that the total execution time has been improved incredibly by average of 32% and 20% (in milliseconds) respectively.

- **Impact of SDDRC on total number of the replications**

SDDRC optimised the total numbers of the replication by a mean of 207.5 across four different available patterns (Sequential access, random access, random walk unitary access, random walk Gaussian access) table 6.4.

Comparing with LRU and LFU, it can be observed that the total execution time has been improved notably by the average of 46% and 47% (in milliseconds ) respectively.

In summation the SDDRC algorithm optimised the total execution time, effective network usage and a number of the replications used for this experiment.

The reason for these improvements can be highlighted as the capability of the SDDRC algorithm to anticipate the replicas with high access probability for future needs of the sites.

## 6.5 Conclusion

In this chapter, a new dynamic replication algorithm has been proposed that applies the anticipatory replication technique.

The algorithm uses the pre-replicating technique for replicating the adjacent files of the requested jobs from different sites. The algorithm consists of three main phases which anticipate the future needs of the sites and pre-replicates the files that no requests have been submitted for them yet.

By pre-replication, sites will have files locally stored, so at the time of need, sites can access the files locally with minimum response time. And as a result, access latency will be decreased and system performance will considerably improve.

To test the performance of the algorithm, the results have been compared with two main replication algorithms: LFU and LRU. Also, three major metrics have been selected as the main evaluation criteria: mean job execution time of all the jobs, the network usage and the total replication numbers.

The algorithms were compared against these metrics in 4 different access patterns; sequential access, random access, random walk unitary access and random walk Gaussian access. We simulated our algorithm using Cloudsim tool. The experimental results show that our proposed algorithm improves the mean job, effective network usage and numbers of the replication needed under the defined access patterns. Having said that the proposed algorithm is using more memory than and it is more complex the existing methods which could be highlighted as disadvantages of this algorithm.

As a future work, it is possible to improve our proposed replication method by not only considering the access patterns catalogue but also by engaging other data mining factors for defining the pre-replication needs. Moreover, the experiments need to use the wide range of data set inputs for analysis.

The replication in real Cloud system are larger than what is used for our experiment, therefore as a future work, the experiment needs to consider the impact of the larger replication procedure when the data structures are more complex. Also, more factors such as fault tolerance and scalability would be considered for replication optimisation.



# Chapter 7

## Conclusion

*“Architecture begins where engineering ends.”- Walter Gropius (1883-1969)*

This chapter highlights the main points of the thesis. It summarises the key concepts of the chapters and elaborates on the contributions, limitations, and future works of the dissertation.

### 7.1 Review

In this thesis, the problems of the load balancing in Cloud Computing have been discussed.

Part one of the thesis presents the research goals. The concept of load balancing in Cloud-based systems is the primary research theme.

In order to establish this theme, in part two of the thesis two load balancing algorithms have been depicted. Minimising the total tasks' execution time is the main objective of these algorithms.

The scheduling algorithms are architected on the mathematical apparatus of the heuristic STEM algorithm and pre-replication strategies. These algorithms could be considered as the novel ways of load balancing in Cloud-based systems.

Applying the STEM algorithm, the key outcomes of the experiment determine the impact of the magnitude and direction of the load on timespan and performance of the system. Moreover, the pre-replication analysis and results explain the importance of the anticipatory behaviour of the system in terms of pre-replicating the high access probable files that can be requested in future by users.

In summary, the conclusion reflects on the accomplishments of the initial aims and the targets explained in chapter 1, along with a comprehensive review on existing load

balancing approaches and mathematical perceptions, respectively, in chapter 2 and chapter 3.

Chapter 4 critiques the importance of the load balancing concept through implementing a research study on HDM Cloud-based systems. The outcomes of the experimental approaches, which have been demonstrated in chapter 5 and chapter 6, determine the effectiveness of the proposed algorithms. Finally, the future work of the thesis has been discussed in this chapter.

## **7.2 Thesis contribution**

- Presented survey of load balancing issues in Cloud Computing.
- Explored and addressed the importance of load balancing concepts in Cloud-based systems through designing and implementing a case study.
- Proposed heuristic algorithm for monitoring the load balancing on workflow applications in Cloud Computing.
- Proposed a novel dynamic pre-replication method which minimises the total execution time and improves the effective network usage.

### **7.2.1 Discussion**

The work presented in this thesis made a significant contribution to the load balancing field by proposing two novel scheduling algorithms.

The thesis first explained the rationale of having a smart load balancing algorithm embedded in the Cloud's architecture. As the numbers of Cloud users are growing rapidly, a balanced Cloud system that ensures high data availability and performance is required. Understanding the requirements of having the balanced system; research questions and the hypothesis of the research were formulated (see chapter 1).

Based on the defined research scope, research papers concerning load balancing were selected and reviewed. The review highlighted the different static and dynamic load balancing algorithms along with common techniques that have been proposed by other researchers across the Cloud platforms.

Focusing on dynamic load balancing algorithms and considering the research questions of the thesis, the review was narrowed down on two popular load balancing approaches; workflow scheduling and replication strategy. Comparing and analysing these two methods, the literature review listed the existing load balancing challenges such as performance and total task execution time issues that needed to be addressed and resolved.

The comprehensive summary of past works identified the load balancing shortcomings and established a clear road map for improving the considered load balancing algorithms (see chapter 2).

Having studied the existing load balancing objectives in the literature review, the thesis proposed two novel methodologies which addressed the load balancing optimisation challenges.

STEM-PSO, the first proposed algorithm, considered the magnitude and direction of the load, while SDDRC applied the pre-replication methodology to optimise the load balancing performance. The thesis analysed the mathematical apparatus of the algorithms with combination of white-box and black-box modelling approaches. The research methodology established a fundamental base for designing and implementing the algorithms through programming and simulations (see chapter 3).

Following the literature review, the thesis tried to apply the reviewed load balancing techniques as a case study, to project the load balancing issues on a real-world Cloud system. HDM, the Cloud-based hospital data management system, was designed and implemented to depict the existing challenges in a Cloud system with the vast numbers of users' requests. Varying test scenarios for the different numbers of the users have been implemented to project the load distribution on HDM. The outcome of the case study clearly identified the path of the thesis (see chapter 4).

Following the research methodology approach, as the first proposed algorithm, the thesis presented a heuristic methodology known as the STEM-PSO algorithm, which incorporates the particle swarm optimisation along with the STEM heuristic scheduling algorithm.

The algorithm employed heuristic techniques to project the magnitude and direction of the existing load between interconnected tasks. The outcomes of the experiment have been compared with the HEFT algorithm. The analysis proved that the proposed load balancing algorithm enhances the availability of the system by minimising the total execution time of the application (see chapter 5).

Taking further enhancement in timespan, the thesis proposed another approach for optimising the load balancing issues through replication strategies. According to the past access catalogue, the proposed method pre-replicates the files with the highest access probability. The outcomes of the experiments have been compared with LRU and LFU replication algorithms. The analysis illustrated that the proposed pre-replication methodology significantly enhanced the total execution time of the applications (see chapter 6).

## 7.2.2 Limitations

The main limitations of this thesis are listed below:

- STEM-PSO experimental simulation needs to consider the large-scale workflow data models.  
The experiment was conducted using workflow applications with the maximum size of 1024. Considering the complexity of the real-world scientific applications such as earthquake monitoring, large-scale data models with more complex structures in terms of connectivity should be considered.
- STEM-PSO should apply other projection models.  
The current experiment considers STEM as the projection operator. Future work, should examine other projection methods such as Lagrangians, which can estimate the magnitude and direction of the load change more accurately in workflow applications.
- STEM-PSO should consider other optimisation models than PSO

The current experiment only considers PSO as an optimisation method. In future work, the experiment may apply the Voronoi partitioning method, which tries to separate the whole workflow application to different sub-workflows. The sub-partitioning could have a major impact on the anticipatory behaviour of the STEM algorithm.

- SDDR, the pre-replication technique, should consider other metrics than past access patterns.

The current experiment only considers the files' past access history to accomplish the pre-replication. As a future work, the experiment could consider more factors such as data mining techniques, which would be a good solution in terms of enhancing the predictability behaviour of the algorithm.

## 7.3 Future work

The future approaches in enhancing the load balancing challenges have been examined below:

1. SLA and load balancing:

The service level agreement identifies the services offered to the customers. If the service provider could not provide the service that was promised, a violation will occur.

For example, if the SLA stated that the service provider will guarantee the minimum response time for VMs and that did not happen, it could be considered as a violation, which may lead to major penalties for the provider.

As the numbers of Cloud users are increasing dramatically, SLA violations are likely to happen due to the load fluctuations and lack of load balancing monitoring.

Therefore, future work on the STEM-PSO load balancing algorithm could include further development to address the following research question:

*“How can the load balancing architecture be enhanced so that Cloud service providers can guarantee their promised SLA and QOS?”*

2. Load balancing based on energy efficiency:

Data centres are expensive to use. They consume carbon footprints which can have a negative impact on the environment. As the numbers of users are increasing, more demand would accrue for building the data centres. In this thesis, we have presented a pre-replication strategy which had a major impact on effective network percentage. Therefore, as a future work from this thesis, the SDDRC could be enhanced to address the following research question:

*“How should the pre-replication strategy be implemented to minimise the energy consumption while balancing the requested tasks in Cloud-based systems?”*

3. Self-awareness behaviour in the workflow scheduling algorithm:

Load balancing optimisation is categorised as NP-hard problems. It plays an important role in enhancing the Cloud utilisation.

Different methods have been proposed to achieve system load balancing in the Cloud environment. VM migration is one of these techniques, which is proposed to improve the functionality of the VMs. Despite the advantages of VM migration, some drawbacks remain, which encourage researchers to improve VM migration methods. Future work is recommended to design an optimised load balancing algorithm that would answer the following research question:

*“How can the self-awareness behaviour between overloaded VMs in Cloud-based systems be best established?”*

The proposed algorithm achieves the system load balancing by applying self-organizing methods between overloaded VMs. This technique is structured based on communications between VMs. It helps the overloaded VMs to transfer their extra tasks to other under-loaded VMs by applying the enhanced feedbacking approach using endocrine methodology.

## 7.4 Final Remarks

The presented load balancing approaches discussed in this thesis showed that the optimised load balancing methods are not limited only to resource allocations and release. As the dissertation validated, analysing the magnitude and direction of the load and anticipating the pre-replicas needed for the Cloud users could have a major impact on enhancing the system's load balancing as well.

Principally, the results in the research action study helped to plan and model the load balancing characteristics of the STEM-PSO and SDDR experiments. Much of this work is already published in the well-cited journals and conference papers. These algorithms are coded through Java programming and I would authorise the other researchers to repeat this code in their research works for further analysis and improvements.

As a final remark, thesis highlights the following outcomes to validate the illustrated research questions:

As a novel solution, STEM-PSO load balancing approach proves that by considering the load fluctuations in workflow applications, the total execution time of the workflow application, memory rate and the CPU usage can be improved.

Moreover, SDDR strategy confirms that the mean job execution time, total numbers of the replicated files and the effective network usage can be revamped greatly by pre-replicating the files with highest access probability.

### **III. Bibliography and publications**



# Chapter 8

## Bibliography

Abraham, R. Buyya and B. Nath, Nature's Heuristics for Scheduling Jobs on Computational Grids, in Proc.

Abrishami, S. & Naghibzadeh, M. 2012. Deadline-constrained workflow scheduling in software as a service Cloud. *Scientia Iranica*, 19, 680-689.

Aceto, G., Botta, A., de Donato, W. & Pescape, A. 2012, 'Cloud monitoring: Definitions, issues and future directions', Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on, pp. 63-7.

Amazon EC2 pricing, viewed on 25 July 2014, available at [www.aws.amazon.com/ec2/pricing](http://www.aws.amazon.com/ec2/pricing). Amazon web services LLC

Amoiralis, E.I., Tsili, M.A. & Kladas, A.G. 2012, 'Global transformer design optimisation using deterministic and non-deterministic algorithms', Electrical Machines (ICEM), 2012 XXth International Conference on, pp. 2323-31.

Anikode, L.R. & Bin, T. 2011, 'Integrating Scheduling and Replication in Data Grids with Performance Guarantee', Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, pp. 1-6.

Annette. R, Banu,A. W and Shriram. Article: A Taxonomy and Survey of Scheduling Algorithms in Cloud: Based on task dependency. *International Journal of Computer Applications* 82(15):20-26, November 2013. Published by Foundation of Computer Science, New York, USA.

Ardagna, D., Casolari, S. & Panicucci, B. 2011, 'Flexible Distributed Capacity Allocation and Load Redirect Algorithms for Cloud Systems', Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp. 163-70.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Zaharia, M., (2009), 'Above the Clouds: A Berkeley View of Cloud Computing', Retrieved from University of California at Berkeley viewd 3 Feb2012<http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf>.

Aslanzadeh, S & Chaczko, Z 2015, 'Generalized Spring Tensor Model: A New Improved Load Balancing Method in Cloud Computing', in H. Selvaraj, D. Zydek & G. Chmaj (eds), *Progress in Systems Engineering*, vol. 330, Springer International Publishing, pp. 831-5.

Atoui, M.A., Verron, S. & Kobi, A. 2015, 'Fault detection with Conditional Gaussian Network', *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 473-81.

Bala,A., & Chana,I. Article: A Survey of Various Workflow Scheduling Algorithms in Cloud Environment. *IJCA Proceedings on 2nd National Conference on Information and Communication Technology NCICT(4):26-30*, November 2011. Published by Foundation of Computer Science, New York, USA.

Barkat, A., dos Santos, A.D. & Thi Thao Nguyen, H. 2014, 'Open Stack and Cloud Stack: Open Source Solutions for Building Public and Private Clouds', *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2014 16th International Symposium on, pp. 429-36.

Barrett, E., Howley, E. & Duggan, J. 2011, 'A Learning Architecture for Scheduling Workflow Applications in the Cloud', *Web Services (ECOWS)*, 2011 Ninth IEEE European Conference on, pp. 83-90.

Batcher, K. W. & Walker, R. A. 2008. Dynamic round-robin task scheduling to reduce cache misses for embedded systems. *Proceedings of the conference on Design, automation and test in Europe - DATE '08*, 260-260.

Belalem, G., Tayeb, F. & Zaoui, W. 2010, 'Approaches to Improve the Resources Management in the Simulator CloudSim', in R. Zhu, Y. Zhang, B. Liu & C. Liu (eds), *Information Computing and Applications*, vol. 6377, Springer Berlin Heidelberg, pp. 189-96.

Bell, W. H., Cameron, D. G., Capozza, L., Millar, A. P., Stockinger, K., and Zini, F. (2002). Simulation of dynamic grid replication strategies in optosim. In *Journal of High Performance Computing Applications*, pages 46–57. Springer-Verlag.

Ben Charrada, F., Ounelli, H. & Chettaoui, H. 2010, 'Dynamic Period vs Static Period in Data Grid Replication', P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on, pp. 565-8.

Bernard. M. W. and Graham R. L. (1989), "The Shortest Network Problem," *Scient@American*, Jan~ary 1989, pp. 84-89.

Bonvin, N., Papaioannou, T.G. & Aberer, K. 2010, 'Cost-efficient and differentiated data availability guarantees in data Clouds', *Data Engineering (ICDE)*, 2010 IEEE 26th International Conference on, pp. 980-3.

Bojanova, I. & Samba, A. 2011, 'Analysis of Cloud Computing Delivery Architecture Models', *Advanced Information Networking and Applications (WAINA)*, 2011 IEEE Workshops of International Conference on, pp. 453-8.

Breiter, G. & Naik, V.K. 2013, 'A Framework for Controlling and Managing Hybrid Cloud Service Integration', *Cloud Engineering (IC2E)*, 2013 IEEE International Conference on, pp. 217-24.

Butz, M., Herbort, O. & Pezzulo, G. 2008, 'Anticipatory, Goal-Directed Behavior', in G. Pezzulo, M. Butz, C. Castelfranchi & R. Falcone (eds), *The Challenge of Anticipation*, vol. 5225, Springer Berlin Heidelberg, pp. 85-113.

Buyya, R., Chee Shin, Y. & Venugopal, S. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. *High Performance Computing and Communications*, 2008. HPCC '08. 10th IEEE International Conference on, 25-27 Sept. 2008 2008. 5-13.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. & Brandic, I. 2009b. Cloud Computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25, 599-616.

Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, S.A.F.D. & Buyya, R. 2011, 'CloudSim: a toolkit for modeling and simulation of Cloud Computing environments and evaluation of resource provisioning algorithms', *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23-50.

Cecelia Clementi, Hugh Nymeyer, and Jose N. Onuchic. Topological and energetic factors: What determines the structural details of the transition state ensemble and en-route intermediates for protein folding? an investigation for small globular proteins. *Journal of Molecular Biology*, 298: 937\_953, 2000.

Chaczko, Z. & Aslanzadeh, S. 2011, 'C2EN: Anisotropic Model of Cloud Computing', *Systems Engineering (ICSEng)*, 2011 21st International Conference on, pp. 467-73.

Chaczko, Z, Mahadevan, V, Aslanzadeh, S, Mcdermid, C, "Availability and load balancing in Cloud Computing", 2011 International conference in computer and software modelling, IACSITPress, Singapore

Chandran, D. & Kempegowda, S. 2010, 'Hybrid E-learning platform based on Cloud architecture model: A proposal', *Signal and Image Processing (ICSIP)*, 2010 International Conference on, pp. 534-7.

Chapman, W.L., Rozenblit, J. & Bahill, A.T. 1994, 'The system design problem is NP-complete', *Systems, Man, and Cybernetics*, 1994. Humans, Information and Technology., 1994 IEEE International Conference on, vol. 2, pp. 1880-4 vol.2.

Chavan, V. & Kaveri, P.R. 2015, 'Shared resource clustering for load balancing and availability in Cloud', *Computing for Sustainable Global Development (INDIACom)*, 2015 2nd International Conference on, pp. 1004-7.

R.S.Chang, H.P.Chang, "A Dynamic Data Replication Strategy Using Access-Weights in Data Grids" *Supercomputing*, Vol 45xz

Chen, C., Liu, J., Wen, Y. & Chen, J. 2015, 'Research on Workflow Scheduling Algorithms in the Cloud', in J. Cao, L. Wen & X. Liu (eds), *Process-Aware Systems*, vol. 495, Springer Berlin Heidelberg, pp. 35-48.

Cheng, W., Xiang-Yang, L., Shaojie, T. & Changjun, J. 2012, 'Capacity and delay tradeoffs in mobile networks under Gaussian channel model', *Mobile Adhoc and Sensor Systems (MASS)*, 2012 IEEE 9th International Conference on, pp. 272-80.

Chiba, T., den Burger, M., Kielmann, T. & Matsuoka, S. 2010, 'Dynamic Load-Balanced Multicast for Data-Intensive Applications on Clouds', *Cluster, Cloud and Grid Computing (CCGrid)*, 2010 10th IEEE/ACM International Conference on, pp.5-14.

Chiu, G.M., Raghavendra, C.S. & Ng, S.M. 1989, 'Resource allocation with load balancing consideration in distributed computing systems', *INFOCOM '89. Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging or Converging*, IEEE, pp. 758-65 vol.2.

Chopra, N. & Singh, S. 2013, 'HEFT based workflow scheduling algorithm for cost optimisation within deadline in hybrid Clouds', *Computing, Communications and Networking Technologies (ICCCNT)*, 2013 Fourth International Conference on, pp. 1-6.

Chun-Chen, H., Chien-Min, W. & Pangfeng, L. 2008, 'Optimal replication transition strategy in distributed hierarchical systems', *Parallel and Distributed Processing*, 2008. IPDPS 2008. IEEE International Symposium on, pp. 1-10.

Conres, IT solutions, 2014, *Cloud Deployment models*, viewed on 10 Feb 2014, <http://www.conres.com/Cloud-computing-deployment-models>.

Cui, L. & Shiyong, L. Scheduling Scientific Workflows Elastically for Cloud Computing. *Cloud Computing (CLOUD)*, 2011 IEEE International Conference on, 4-9 July 2011 2011. 746-747.

Daniel M. Dubois. Introduction to computing anticipatory systems. *International Journal of ComputingAnticipatory Systems*, 2:3\_14, 1998.

Das, S.K., Pinotti, M.C. & Sarkar, F. 1996, 'Optimal and load balanced mapping of parallel priority queues in hypercubes', *Parallel and Distributed Systems*, IEEE Transactions on, vol. 7, no. 6, pp. 555-64.

Devine, K. D., Boman, E. G., Heaphy, R. T., Hendrickson, B. A., Teresco, J. D., Faik, J., Flaherty, J. E. & Gervasio, L. G. 2005. New challenges in dynamic load balancing. *Applied Numerical Mathematics*, 52, 133-152.

Dobber, M., van der Mei, R. & Koole, G. 2009, 'Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison', *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 207-18.

Domanal, S.G. & Reddy, G.R.M. 2014, 'Optimal load balancing in Cloud Computing by efficient utilisation of virtual machines', *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, pp. 1-4.

Dongjin, Y. & Kwang-Mong, S. A scheduling mechanism for multiple MapReduce jobs in a workflow application (position paper). *Computing, Communications and Applications Conference (ComComAp), 2012, 11-13 Jan. 2012* 2012. 405-410.

Dong-Sheng, Y., Yin-Long, L., Zhong, L. & Wei-Ming, Z. 2004, 'Research on algorithms of task scheduling', *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 1, pp. 42-7 vol.1.

DTU, Department of Applied mathematics and computer science, 'Gray Box', Viewd on 20 September 2015, <http://energy.imm.dtu.dk/models/gray-box.html>

E. Deelman and A. Chervenak, "Data Management Challenges of Data-Intensive Scientific Workflows," in *CCGRID '08: Proceedings of the Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. Washington, DC, USA:IEEE, 2008, pp. 687–692.

E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G.Mehta, K. Vahi, K. Blackburn, A. Lazzarini,A. Arbree, R. Cavanaugh, and S. Koranda, "Mapping Abstract Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. V1, no. 1, pp. 25–39, March 2003.

E. M. Bahsi, E. Ceyhan, and T. Kosar, "Conditional workflow management: A survey and analysis," *Scientific Programming*, vol. 15, no. 4, pp. 283–297, 2007.

- Erzberger, C. 2001, 'Bernard, H. Russell: Social research methods. Qualitative and quantitative approaches', *KZfSS Kölner Zeitschrift für Soziologie und Sozialpsychologie*, vol. 53, no. 4, pp. 804-6.
- Fiandrino, C., Kliazovich, D., Bouvry, P. & Zomaya, A.Y. 2015, 'Performance and Energy Efficiency Metrics for Communication Systems of Cloud Computing Data Centers', *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1-.
- Gaikwad-Patil, T. 2012. *Load Balancing The Essential Factor In Cloud Computing*. 1, 1-5.
- Galante, G. and L. C. E. de Bona (2012). A Survey on Cloud Computing Elasticity. *Utility and Cloud Computing (UCC)*, 2012 IEEE Fifth International Conference on.
- Gang, S., Anand, V., Hong-Fang, Y., Dan, L. & Lemin, L. 2012, 'Optimal provisioning for elastic service oriented virtual network request in Cloud Computing', *Global Communications Conference (GLOBECOM)*, 2012 IEEE, pp. 2517-22.
- Georgi Dalakov, *History of computer, software, hardware, Internet*, California, viewed 12 September 2015, < <http://history-computer.com/>>
- Ghilavizadeh Zeinab, seyed javad mirabedini, Ali Harounabadi , 'A new Fuzzy optimal data Replication for data Grid', *Management science*, pp.927-936, 2013.
- Ghutke, B. & Shrawankar, U. 2014, 'Pros and cons of load balancing algorithms for Cloud Computing', *Information Systems and Computer Networks (ISCON)*, 2014 International Conference on, pp. 123-7.
- Gibson, J., Rondeau, R., Eveleigh, D. & Qing, T. 2012, 'Benefits and challenges of three Cloud Computing service models', *Computational Aspects of Social Networks (CASoN)*, 2012 Fourth International Conference on, pp. 198-205.
- Gopularam, B.P., Yogeesh, C.B. & Periasamy, P. 2012, 'Highly scalable model for tests execution in Cloud environments', *Advanced Computing and Communications (ADCOM)*, 2012 18th Annual International Conference on, pp. 54-8.

- Gonzalez, A.J. & Helvik, B.E. 2012, 'System management to comply with SLA availability guarantees in Cloud Computing', *Cloud Computing Technology and Science (CloudCom)*, 2012 IEEE 4th International Conference on, pp. 325-32.
- Grant, A.B. & Eluwole, O.T. 2013, 'Cloud resource management &#x2014; Virtual machines competing for limited resources', *ELMAR*, 2013 55th International Symposium, pp. 269-74.
- Greenwood, G.W. 2001, 'Finding solutions to NP problems: philosophical differences between quantum and evolutionary search algorithms', *Evolutionary Computation*, 2001. Proceedings of the 2001 Congress on, vol. 2, pp. 815-22 vol. 2.
- Grubitzsch, P. & Schuster, D. 2014, 'Hosting and Discovery of Distributed Mobile Services in an XMPP Cloud', *Mobile Services (MS)*, 2014 IEEE International Conference on, pp. 47-54.
- Guang Song. Spring tensor model source code, 2012. URL <http://www.cs.iastate.edu/~gsong/CSB/>.
- Guicheng, S. & Bingwu, L. 2010, 'Research on Application of Internet of Things in Electronic Commerce', *Electronic Commerce and Security (ISECS)*, 2010 Third International Symposium on, pp. 13-6.
- Gupta, A., et al. (2013). Improving HPC Application Performance in Cloud through Dynamic Load Balancing. *Cluster, Cloud and Grid Computing (CCGrid)*, 2013 13th IEEE/ACM International Symposium on.
- Gupta, A., Sarood, O., Kale, L.V. & Milojcic, D. 2013, 'Improving HPC Application Performance in Cloud through Dynamic Load Balancing', *Cluster, Cloud and Grid Computing (CCGrid)*, 2013 13th IEEE/ACM International Symposium on, pp. 402-9.
- Hao, L., Binxing, F. & Xiaochun, Y. 2006, 'Anomaly Detection in SMTP Traffic', *Information Technology: New Generations*, 2006. ITNG 2006. Third International Conference on, pp. 408-13.



Haozheng, R., Yihua, L. & Chao, Y. 2012, 'The load balancing algorithm in Cloud Computing environment', *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, pp. 925-8.

Haozheng, R., Yihua, L. & Chao, Y. The load balancing algorithm in Cloud Computing environment. *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, 29-31 Dec. 2012 2012. 925-928.

Hasan, M.Z., Magana, E., Clemm, A., Tucker, L. & Gudreddi, S.L.D. 2012, 'Integrated and autonomic Cloud resource scaling', *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pp. 1327-34.

Hataba, M. & El-Mahdy, A. 2012, 'Cloud Protection by Obfuscation: Techniques and Metrics', *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*, pp. 369-72.

He, D.-X. & Jia, R.-m. 2012, 'Cloud model-based Artificial Bee Colony Algorithm's application in the logistics location problem', *Information Management, Innovation Management and Industrial Engineering (ICIII), 2012 International Conference on*, vol. 1, pp. 256-9.

H. Belhaddad, R. Perez, M. Nicolaidis, R. Gaillard, M. Derby, F. Benistant, "Circuit Simulations of SEU and SET Disruptions by Means of an Empirical Model Built Thanks to a Set of 3D MixedMode Device Simulation Responses", *Proceedings of RADECS, 2006*

Hayashi, Y., Spencer, M.C. & Nasuto, S.J. 2013, 'A study of anticipatory non-autonomous systems', *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pp. 316-8.

Hong, H. 2010, 'Applications deployment on the SaaS platform', *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, pp. 232-7.

Hongli, Z., Lin, Y., Xiaojiang, D. & Guizani, M. 2013, 'Protecting private Cloud located within public Cloud', *Global Communications Conference (GLOBECOM), 2013 IEEE*, pp. 677-81.

- Hornsby, A. & Walsh, R. 2010, 'From instant messaging to Cloud Computing, an XMPP review', Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on, pp. 1-6.
- Hoyer, P. 1995, 'A general technique for implementation of efficient priority queues', Theory of Computing and Systems, 1995. Proceedings., Third Israel Symposium on the, pp. 57-66.
- Huichao, M. & Yongqiang, Y. 2012, 'An Effective Clustering-Based Prefetching Scheme for Spatial Databases System', Internet Computing for Science and Engineering (ICICSE), 2012 Sixth International Conference on, pp. 35-41.
- Kitajima, N., Goto, Y. & Jingde, C. 2008, 'Fast Qualitative Reasoning about Actions for Computing Anticipatory Systems', Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, pp. 171-8.
- I. Brandic, S. Pllana, and S. Benkner, "Specification, planning, and execution of QoS-aware Grid workflows within the Amadeus environment," *Concurrency and Computation: Practice & Experience*, vol. 20, pp. 331–345, March 2008.
- I. Foster, K. Ranganathan, design and evaluation of dynamic replication strategies a high performance data grid, in :proceedings of international conference on computing in High energy and nuclear physics, china, September 2001.
- Iskander, M.K., Trainor, T., Wilkinson, D.W., Lee, A.J. & Chrysanthis, P.K. 2014, 'Balancing Performance, Accuracy, and Precision for Secure Cloud Transactions', Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 2, pp. 417-26.
- Jae Yoo, L. & Soo Dong, K. 2010, 'Software Approaches to Assuring High Scalability in Cloud Computing', e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on, pp. 300-6.
- Jain, A., Yadav, A., Namboodiri, L. & Abraham, J. 2013, 'A Threshold Band Based Model for Automatic Load Balancing in Cloud Environment', Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on, pp. 1-7.

Jihoon, K., Yohan, K., Jongwon, L., Yongjoo, K., Hwisoo, S., Kyoungwoo, L. & Yunheung, P. 2013, 'Selective validations for efficient protections on Coarse-Grained Reconfigurable Architectures', *Application-Specific Systems, Architectures and Processors (ASAP)*, 2013 IEEE 24th International Conference on, pp. 95-8.

Jenn-Wei, L., Chien-Hung, C. & Chang, J.M. 2013, 'QoS-Aware Data Replication for Data-Intensive Applications in Cloud Computing Systems', *Cloud Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 101-15.

J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task scheduling strategies for workflow-based applications in grids," in *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 2*. Washington, DC, USA: IEEE, 2005, pp. 759–767.

J. Broberg, R. Buyya, and Z. Tari, "MetaCDN: Harnessing 'storage Clouds' for high performance content delivery," *Journal of Network and Computer Applications*, vol. 32, no. 5, pp. 1012–1022, 2009.

J. Griffioen, R. Appleton, Performance measurements of automatic prefetching, *Parallel and Distributed Computing Systems* (1995) 165–170.

J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Journal of Grid Computing*, vol. 3, no. 3, pp. 171–200, September 2005.

J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow Scheduling Algorithms for Grid Computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*, ser. Studies in Computational Intelligence. Springer Berlin / Heidelberg, 2008, vol. 146, pp. 173–214.

Jianzong, W., Rui, H., Yifeng, Z., Jiguang, W., Changsheng, X. & Yanjun, C. 2012, 'RO-BURST: A Robust Virtualisation Cost Model for Workload Consolidation over Clouds', *Cluster, Cloud and Grid Computing (CCGrid)*, 2012 12th IEEE/ACM International Symposium on, pp. 490-7.

- JungYul, C., Seung-Hoon, K., Mi-Jeong, L., Taeil, C., Byoung-Kwon, S. & Jae-Hyoung, Y. 2010, 'Service traffic management system for multiservice IP networks: lessons learned and applications', *Communications Magazine, IEEE*, vol. 48, no. 4, pp. 58-65.
- Junjie, T., Renjie, P. & Ke, X. 2011, 'Cloud Computing infrastructure based on named content', *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pp. 429-34.
- K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a modified BHR region based algorithm, *Future Generation Computer Systems* 27 (2011),202–210.
- Kadim, H.J. 2007, 'Modelling of Anticipatory Behaviour for Self-Control and Adaptability with Applications to Autonomous Systems', *Bio-inspired, Learning, and Intelligent Systems for Security, 2007. BLISS 2007. ECSIS Symposium on*, pp. 91-6.
- Kalimeri, M., Derreumaux, P. & Sterpone, F. 2015, 'Are coarse-grained models apt to detect protein thermal stability? The case of OPEP force field', *Journal of Non-Crystalline Solids*, vol. 407, pp. 494-501.
- Karim, R., Chen, D. & Miri, A. 2015, 'End-to-End Performance Prediction for Selecting Cloud Services Solutions', *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*, pp. 69-77.
- Kamra, M. & Manna, R. 2012, 'Performance of Cloud-Based Scalability and Load with an Automation Testing Tool in Virtual World', *Services (SERVICES), 2012 IEEE Eighth World Congress on*, pp. 57-64.
- Kennedy, J.Eberhart, R: Particle swarm optimisation In: *IEEE International Conference on Neutral Network*, PP.1942-1948 (1995)
- Kimura, S., Sonoda, K., Yamane, S., Matsumura, K. & Hatakeyama, M. 2006, 'Function Approximation Approach to the Inference of Normalized Gaussian Network Models of Genetic Networks', *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pp. 2218-25.

Kozlovsky, M., Trocsik, M., Schubert, T. & Poserne, V. 2013, 'IaaS type Cloud infrastructure assessment and monitoring', *Information & Communication Technology Electronics & Microelectronics (MIPRO)*, 2013 36th International Convention on, pp. 249-52.

Kpmg Cutting through complexity 2011, *The Cloud: Changing the Business Ecosystem*, KPMG, India, viewed 2 Feb 2014 <[http://www.kpmg.com/IN/en/IssuesAndInsights/ThoughtLeadership/The\\_Cloud\\_Changing\\_the\\_Business\\_Ecosystem.pdf](http://www.kpmg.com/IN/en/IssuesAndInsights/ThoughtLeadership/The_Cloud_Changing_the_Business_Ecosystem.pdf).

Kruber, N., Ho, X., Gqvist, M. & Schutt, T. 2011, 'The Benefits of Estimated Global Information in DHT Load Balancing', *Cluster, Cloud and Grid Computing (CCGrid)*, 2011 11th IEEE/ACM International Symposium on, pp. 382-91.

K. Tanaka, H. Nakamura, T. Uemura, K. Takeuchi, T. Fukuda, S. Kumashiro, "Study on Influence of Device Structure Dimensions and Profiles on Charge Collection Current Causing SET Pulse Leading to Soft Errors in Logic Circuits", *SISPAD 2009*

Kun, L., Gaochao, X., Guangyu, Z., Yushuang, D. & Wang, D. *Cloud Task Scheduling Based on Load Balancing Ant Colony Optimisation. Chinagrid Conference (ChinaGrid)*, 2011 Sixth Annual, 22-23 Aug. 2011 2011. 3-9.

Lai Gong, G., You Rui, H., Jun, C. & Li Guo, Q. 2011, 'Investigation of architecture, key technology and application strategy for the Internet of things', *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC)*, 2011, vol. 2, pp. 1196-9.

Lamb, J. & Cusato, T. 1994, 'LAN-based office for the enterprise, a case study', *Local Computer Networks*, 1994. Proceedings., 19th Conference on, pp. 440-7.

Lamehamedi.H and Szymanski.B, 'Decentralized data management framework for data grids' *Future Generation computer system*, pp.109-115, 2007.

Lazri, K., Laniepce, S., Zheng, H. & Ben-Othman, J. 2014, 'AMAD: Resource Consumption Profile-Aware Attack Detection in IaaS Cloud', *Utility and Cloud Computing (UCC)*, 2014 IEEE/ACM 7th International Conference on, pp. 379-86.

Lee, B., Grance, T., Patt-Corner, R. & Voas, J 2012, Cloud Computing Synopsis and Recommendations, viewed on 15 September 2014, <http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf>

Lee, K., Paton, N.W., Sakellariou, R., Deelman, E., Fernandes, A.A.A. & Mehta, G. 2008, 'Adaptive Workflow Processing and Execution in Pegasus', Grid and Pervasive Computing Workshops, 2008. GPC Workshops '08. The 3rd International Conference on, pp. 99-106.

Lee, R. and J. Bingchiang (2011). Load-Balancing Tactics in Cloud. Cyber-Enabled Distributed computing and Knowledge Discovery (CyberC), 2011 International Conference on.

Lei M, Vrbsky S V, Hong X, 'An on-line replication strategy to increase availability in data grids'. Future Generation Computer Systems, 24(2) .pp 85-98, 2008.

Leifur Leifsson, Hildur Saevarsdottir, Sven Sigurosson, and Ari Vesteinsson. Gray-box modelling of an ocean vessel for operational optimisation. Simulation Modelling Practice and Theory, 16:923932, 2008. doi: 10.1016/j.simpat.2008.03.006.

Li, Y, & Mascagni, M 2012, ", North Carolina, viewed on Feb 6th 2013, [http://www.cs.odu.edu/~yaohang/publications/IADIS\\_Li.pdf](http://www.cs.odu.edu/~yaohang/publications/IADIS_Li.pdf).

Liu, X., Yuan, D., Zhang, G., Li, W., Cao, D., He, Q., Chen, J. & Yang, Y. 2012. Case Study: SwinDeW-C Cloud Workflow System. The Design of Cloud Workflow Systems. Springer New York.

Loukopoulos, T. & Ahmad, I. 2000, 'Static and adaptive data replication algorithms for fast information access in large distributed systems', Distributed Computing Systems, 2000. Proceedings. 20th International Conference on, pp. 385-92.

Luo, H.-m., Yan, C.-k. & Luo, J.-w. 2012, 'Dynamic Programming Based Grid Workflow Scheduling Algorithm', in Y. Wu (ed.), Software Engineering and Knowledge Engineering: Theory and Practice, vol. 114, Springer Berlin Heidelberg, pp. 993-1000.

- M. Shorfuzzaman, P. Graham, R. Eskicioglu, Popularity-driven dynamic replica placement in hierarchical data grids, in: Proceedings of Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, pp. 524–531.
- M. Tang, B.S. Lee, C.K. Yao, X.Y. Tang, Dynamic replication algorithm for the multi-tier data grid, *Future Generation Computer Systems* 21 (5) (2005).775–790.
- Madivi, R. & Kamath, S.S. 2014, 'An hybrid bio-inspired task scheduling algorithm in Cloud environment', *Computing, Communication and Networking Technologies (ICCCNT)*, 2014 International Conference on, pp. 1-7.
- Maguluri, S.T., Srikant, R. & Lei, Y. 2012, 'Stochastic models of load balancing and scheduling in Cloud Computing clusters', *INFOCOM, 2012 Proceedings IEEE*, pp. 702-10.
- Manish, M. & Cheema, R. K. A Min-Max Approach to Perform Load Balancing in Distributed Network. *Advanced Computing & Communication Technologies (ACCT)*, 2012 Second International Conference on, 7-8 Jan. 2012 2012. 200-203.
- Maarouf, A., Marzouk, A. & Haqiq, A. 2014, 'Automatic control of the quality of service contract by a third party in the Cloud Computing', *Complex Systems (WCCS)*, 2014 Second World Conference on, pp. 599-603.
- Masiyev, K.H., Qasymov, I., Bakhishova, V. & Bahri, M. 2012, 'Cloud Computing for business', *Application of Information and Communication Technologies (AICT)*, 2012 6th International Conference on, pp. 1-4.
- Mansouri, N., Dastghaibfard, G.H. & Mansouri, E. 2013, 'Combination of data replication and scheduling algorithm for improving data availability in Data Grids', *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 711-22.
- Mathur,P and Nishchal,N “Cloud Computing: New challenge to the entire computer industry,” in *Parallel Distributed and Grid Computing (PDGC)*, 2010 1st International Conference on, 2010, pp. 223–228.

Maurice Clerc. Particle Swarm Optimisation - L'Optimisation par Essaims Particulaires. Wiley ISTE,London, England, 2006.

Metcalfe, B. 2000, 'The next-generation Internet', *Internet Computing*, IEEE, vol. 4, no. 1, pp. 58-9.

McCarthy, John. "The Home Information Terminal." *Man and Computer*, Proc. Int. Conf., Bordeaux 1970, pp. 48-57. Basel: Karger, 1972. Anticipated the usefulness of computer access from the home, much of which would not be realized until the appearance of networked personal computers in the early 1980s.

Mell,P & Grance,T .2011, The NIST Definition of Cloud Computing, viewd 19 September 2014, <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>

Metwally, K.M., Jarray, A. & Karmouch, A. 2015, 'Two-phase ontology-based resource allocation approach for IaaS Cloud service', *Consumer Communications and Networking Conference (CCNC)*, 2015 12th Annual IEEE, pp. 790-5.

Miyake, Y., Onishi, Y. & Popper, E. 2003, 'Two types of anticipation in sensory-motor coupling', *SICE 2003 Annual Conference*, vol. 1, pp. 551-6 Vol.1.

Ming-Ke, L. & Yaw-Chung, C. 2014, 'An XMPP-Based XML Representation Middleware to Build Universal Service-Oriented Gateway in M2M Environment', *Internet of Things (iThings)*, 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing(CPSCom), IEEE, pp. 193-200.

Ming, Z. & Zu-Kuan, W. 2010, 'Load-balancing strategies in distributed workflow management system', *Communication Systems, Networks and Applications (ICCSNA)*, 2010 Second International Conference on, vol. 1, pp. 235-8.

Moreno-Vozmediano, R., et al. (2013). "Key Challenges in Cloud Computing: Enabling the Future Internet of Services." *Internet Computing*, IEEE17(4): 18-25.

Nadeem, F., Prodan, R. & Fahringer, T. 2008, 'Characterizing, Modelling and Predicting Dynamic Resource Availability in a Large Scale Multi-purpose Grid',



Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on, pp. 348-57.

Nader-uz-zaman, M., Kashem, M.A., Ahmad, R.B. & Rahman, M. 2014, 'An adaptive replication model for heterogeneous systems', Electronic Design (ICED), 2014 2nd International Conference on, pp. 58-63.

Naik, V.K., Beaty, K. & Kundu, A. 2014, 'Service Usage Metering in Hybrid Cloud Environments', Cloud Engineering (IC2E), 2014 IEEE International Conference on, pp. 253-60.

Nguyen, S.B.S. & Mengjie, Z. 2014, 'A hybrid discrete particle swarm optimisation method for grid computation scheduling', Evolutionary Computation (CEC), 2014 IEEE Congress on, pp. 483-90.

Nurain, N., Sarwar, H., Sajjad, M.P. & Mostakim, M. 2012, 'An In-depth Study of Map Reduce in Cloud Environment', Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on, pp. 263-8.

Ozcan, I. & Bora, S. 2011, 'A hybrid load balancing model for multi-agent systems', Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on, pp. 182-7.

P. Couvares, T. Kosar, A. Roy, J. Weber, and K. Wenger, "Workflow Management in Condor," in *Workflows for e-Science*, I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds. Springer London, 2007, pp. 357–375.

Pandey, S., Linlin, W., Guru, S. M. & Buyya, R. A Particle Swarm Optimisation-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 20-23 April 2010 2010. 400-407.

Patel, G. 2012, 'DICOM Medical Image Management the challenges and solutions: Cloud as a Service (CaaS)', Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on, pp. 1-5.

Pattanaik, P.A., Roy, S. & Pattnaik, P.K. 2015, 'Performance study of some dynamic load balancing algorithms in Cloud Computing environment', Signal Processing and Integrated Networks (SPIN), 2015 2nd International Conference on, pp. 619-24.

Pawara, S.R. & Hiray, S.R. 2013, 'Instant Notification System in Heterogeneous Sensor Network with Deployment of XMPP Protocol', Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on, pp. 87-92.

Peng, L. & Ravindran, B. 2004, 'Fast, best-effort real-time scheduling algorithms', Computers, IEEE Transactions on, vol. 53, no. 9, pp. 1159-75.

Pezzulo, G., Butz, M. & Castelfranchi, C. 2008, 'The Anticipatory Approach: Definitions and Taxonomies', in G. Pezzulo, M. Butz, C. Castelfranchi & R. Falcone (eds), The Challenge of Anticipation, vol. 5225, Springer Berlin Heidelberg, pp. 23-43.

Philippe Kruchten, Henk Obbink, and Judith Staord. The past, present and future of software architecture. IEEE Software, 23:22\_30, 2006.

Pius, S.V. & Suresh, S. 2015, 'A novel algorithm of load balancing in distributed file system for Cloud', Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on, pp. 1-4.

Prabavathy, B., Priya, K. & Babu, C. 2013, 'A load balancing algorithm for private Cloud storage', Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on, pp. 1-6.

Qingling, W. & Varela, C.A. 2011, 'Impact of Cloud Computing Virtualisation Strategies on Workloads' Performance', Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, pp. 130-7.

Radojevic, B. & Zagar, M. Analysis of issues with load balancing algorithms in hosted (Cloud) environments. MIPRO, 2011 Proceedings of the 34th International Convention, 23-27 May 2011 2011. 416-420.

Rahman, M., Venugopal, S. & Buyya, R. A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids. e-Science and Grid Computing, IEEE International Conference on, 10-13 Dec. 2007 2007. 35-42.

Rahman, M., Iqbal, S. & Gao, J. 2014, 'Load Balancer as a Service in Cloud Computing', *Service Oriented System Engineering (SOSE)*, 2014 IEEE 8th International Symposium on, pp. 204-11.

Rajalakshmi, A., Vijayakumar, D. & Srinivasagan, K.G. 2014, 'An improved dynamic data replica selection and placement in Cloud', *Recent Trends in Information Technology (ICRTIT)*, 2014 International Conference on, pp. 1-6.

Ramaswamy, P. 2009 : Signature-driven load management for Cloud Computing infrastructures. 2009 17th International Workshop on Quality of Service, 1-9.

Ramezani, F., Lu, J. & Hussain, F. 2013, "A Fuzzy Predictable Load Balancing Approach in Cloud Computing", *The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, Athens, pp. 108.

Randles, M., Lamb, D. & Taleb-Bendiab, A. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. *Advanced Information Networking and Applications Workshops (WAINA)*, 2010 IEEE 24th International Conference on, 20-23 April 2010 2010. 551-556.

Rimal, B.P., Eunmi, C. & Lumb, I. 2009, 'A Taxonomy and Survey of Cloud Computing Systems', *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pp. 44-51.

Robert Rosen. *Dynamical Systems Theory in Biology*. Wiley Interscience, New York, 1970.

Rosen, R. 1991, 'Anticipatory Systems in Retrospect and Prospect', *Facets of Systems Science*, vol. 7, Springer US, pp. 537-57.

Ruay-shiung chang.Hui-Ping Chang , 'A dynamic data replication strategy using access weights in data grids, *springer Science+Business media*, pp.278-294, 2008.

Ruixia, T. & Xiongfeng, Z. 2010, 'A Load Balancing Strategy Based on the Combination of Static and Dynamic', *Database Technology and Applications (DBTA)*, 2010 2nd International Workshop on, pp. 1-4.

- Sahu, Y., Pateriya, R.K. & Gupta, R.K. 2013, 'Cloud Server Optimisation with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm', Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on, pp. 527-31.
- Saint-Andre, P. 2009, 'XMPP: lessons learned from ten years of XML messaging', Communications Magazine, IEEE, vol. 47, no. 4, pp. 92-6.
- Sakellariou, R. & Henan, Z. A Hybrid heuristic for DAG scheduling on heterogeneous systems. Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 26-30 April 2004 2004. 111.
- Salapura, V. 2012, 'Cloud Computing: Virtualisation and resiliency for data center computing', Computer Design (ICCD), 2012 IEEE 30th International Conference on, pp. 1-2.
- Saravanakumar, C. & Arun, C. 2012, 'Instance management for software application of the Cloud over common deployment model', Emerging Trends in Science, Engineering and Technology (INCOSSET), 2012 International Conference on, pp. 63-7.
- Sarmila, G.P., Gnanambigai, N. & Dinadayalan, P. 2015, 'Survey on fault tolerant &#x2014; Load balancing algorithms in Cloud Computing', Electronics and Communication Systems (ICECS), 2015 2nd International Conference on, pp. 1715-20.
- Sawant, S. 2011. A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment.
- Selimi, M., Freitag, F., Pueyo Centelles, R. & Moll, A. 2014, 'Distributed Storage and Service Discovery for Heterogeneous Community Network Clouds', Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on, pp. 204-12.
- Shahapure, N.H. & Jayarekha, P. 2014, 'Load Balancing with Optimal Cost Scheduling Algorithm', Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference on, pp. 24-31.
- Shameem, P. M. & Shaji, R. S. 2013. A Methodological Survey on Load Balancing Techniques in Cloud Computing. 5, 3801-3812.

Shanjiang, T., Bu-Sung, L. & Bingsheng, H. 2014, 'Towards Economic Fairness for Big Data Processing in Pay-as-You-Go Cloud Computing', Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, pp. 638-43.

Shao, X., Jibiki, M., Teranishi, Y. & Nishinaga, N. 2015, 'Fast and Cost-Effective Load Balancing Method for Range Queriable Cloud Storage', Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual, vol. 3, pp. 638-9.

Shaw, S.B. & Singh, A.K. 2014, 'A survey on scheduling and load balancing techniques in Cloud Computing environment', Computer and Communication Technology (ICCCT), 2014 International Conference on, pp. 87-95.

Shicong, M. & Ling, L. 2013, 'Enhanced Monitoring-as-a-Service for Effective Cloud Management', Computers, IEEE Transactions on, vol. 62, no. 9, pp. 1705-20.

Shobana, G., Geetha, M. & Suganthe, R.C. 2014, 'Nature inspired preemptive task scheduling for load balancing in Cloud datacenter', Information Communication and Embedded Systems (ICICES), 2014 International Conference on, pp. 1-6.

Shu-Ching, W., Kuo-Qin, Y., Wen-Pin, L. & Shun-Sheng, W. Towards a Load Balancing in a three-level Cloud Computing network. 2010. 108-113.

Shuang, C., Ghorbani, M., Yanzhi, W., Bogdan, P. & Pedram, M. 2014, 'Trace-Based Analysis and Prediction of Cloud Computing User Behavior Using the Fractal Modeling Technique', Big Data (BigData Congress), 2014 IEEE International Congress on, pp. 733-9.

Singh, L & Singh, S. Article: A Survey of Workflow Scheduling Algorithms and Research Issues. International Journal of Computer Applications 74(15):21-28, July 2013. Published by Foundation of Computer Science, New York, USA.

Sinititskiy, A.V. & Voth, G.A. 2013, 'Coarse-graining of proteins based on elastic network models', Chemical Physics, vol. 422, pp. 165-74.

Soni, G. & Kalra, M. 2014, 'A novel approach for load balancing in Cloud data center', Advance Computing Conference (IACC), 2014 IEEE International, pp. 807-12.

Sreenivas, V., Prathap, M. & Kemal, M. 2014, 'Load balancing techniques: Major challenge in Cloud Computing - a systematic review', Electronics and Communication Systems (ICECS), 2014 International Conference on, pp. 1-6.

Sqalli, M.H. & Sirajuddin, S. 2005, 'An adaptive load-balancing approach to XML-based network management using JPVM', Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on, vol. 1, p. 6 pp.

Suciu, G., Halunga, S., Vulpe, A. & Suciu, V. 2013, 'Generic platform for IoT and Cloud Computing interoperability study', Signals, Circuits and Systems (ISSCS), 2013 International Symposium on, pp. 1-4.

Suresh, P., Daniel, J.V., Parthasarathy, V. & Aswathy, R.H. 2014, 'A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment', Science Engineering and Management Research (ICSEMR), 2014 International Conference on, pp. 1-8.

Swarnkar, N. 2013. A Survey of Load Balancing Techniques in Cloud Computing. 2, 800-804.

Tang, L. & Chen, H. 2014, 'Joint Pricing and Capacity Planning in the IaaS Cloud Market', Cloud Computing, IEEE Transactions on, vol. PP, no. 99, pp. 1-.

T. A. Feo and M. G. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimisation*, vol. 6, no. 2, pp. 109–133, March 1995.

T. D. Braun, H. J. Siegel, N. Beck, L. L. B'ol'oni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics formapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

T.M. Kroegar, D.E. Long Darrell, The case for efficient file access pattern modelling, in: Proceedings of the 7th Workshop on Hot Topics in Operating Systems, Rio Risco, USA, March 1999, pp. 14–19.

Tilak.D and Patil.P, “A Survey of Various Scheduling Algorithms in Cloud Environment,” vol. 1, no. 2, pp. 36–39, 2012.

Tu-Liang Lin and Guang Song. Generalized spring tensor models for protein fluctuation dynamics and conformation changes. *BMC Structural Biology*, 10:12, 2010. doi: 10.1186/1472-6807-10-S1-S3.

Tungnguyen,AnthonyCutway,andWeisongShi,”Differentiated Replication strategy in data centers, *IFIP international federation of information processing* , pp.276-287, 2010.

Undheim, A., Chilwan, A. & Heegaard, P. 2011, 'Differentiated Availability in Cloud Computing SLAs', *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, pp. 129-36.

Vande Sande, H. & Hameyer, K. 2002, 'Comparison of neural network and polynomial models for the approximation of nonlinear and anisotropic ferromagnetic materials', *Science, Measurement and Technology, IEE Proceedings -*, vol. 149, no. 5, pp. 214-7.

Varalakshmi, P., Ramaswamy, A., Balasubramanian, A. & Vijaykumar, P. 2011. An Optimal Workflow Based Scheduling and Resource Allocation in Cloud.

Venkatesan, R. & Solomi, M.B. 2011, 'Analysis of Load Balancing Techniques in Grid', in V. Das & N. Thankachan (eds), *Computational Intelligence and Information Technology*, vol. 250, Springer Berlin Heidelberg, pp. 147-51.

Vijindra & Shenai, S. 2012. Survey on Scheduling Issues in Cloud Computing. *Procedia Engineering*, 38, 2881-2888.

Vilutis, G., Daugirdas, L., Kavaliunas, R., Sutiene, K. & Vaidelys, M. 2012, 'Model of load balancing and scheduling in Cloud Computing', *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on*, pp. 117-22.

Vu, Q., Lupu, M. & Ooi, B. 2010, 'Load Balancing and Replication', *Peer-to-Peer Computing*, Springer Berlin Heidelberg, pp. 127-56.

- Wan, G., Dai, X., Yin, Q., Shi, X. & Qiao, Y. 2015, 'Interaction of menthol with mixed-lipid bilayer of stratum corneum: A coarse-grained simulation study', *Journal of Molecular Graphics and Modelling*, vol. 60, pp. 98-107.
- Wang, N., Yang, Y., Meng, K., Chen, Y. & Ding, H. 2013, 'A task scheduling algorithm based on qos and complexity-aware optimisation in Cloud Computing', *Information and Communications Technology 2013, National Doctoral Academic Forum on*, pp. 1-8.
- Wang, N. & Xu, D. 2008, 'Resource summary for pay-as-you-go dataspace systems', *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pp. 2842-5.
- Wang, Y., Bahati, R. M. & Bauer, M. A. 2011. A novel deadline and budget constrained scheduling heuristics for computational grids. *Journal of Central South University of Technology*, 18, 465-472.
- Wegscheider, F., Bessler, S. & Gruber, G. 2005, 'Interworking of presence protocols and service interfaces', *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, vol. 4, pp. 45-52 Vol. 4.
- Wei-Tek, T., Xin, S. & Balasooriya, J. 2010, 'Service-Oriented Cloud Computing Architecture', *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pp. 684-9.
- Weixiong, R., Chao, C., Pan, H. & Tarkoma, S. 2013, 'Replication-Based Load Balancing in Distributed Content-Based Publish/Subscribe', *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pp. 765-74.
- Wenbo, Z., Xiang, H., Ningjiang, C., Wei, W. & Hua, Z. 2012, 'PaaS-Oriented Performance Modeling for Cloud Computing', *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pp. 395-404.
- Wenhong, T., Yong, Z., Yuanliang, Z., Minxian, X. & Chen, J. 2011, 'A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters', *Cloud*



Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, pp. 311-5.

Wesam Elshamy. Particle swarm optimisation source code, 2012. URL <http://www.mathworks.com/matlabcentral/fileexchange/authors/24384/>.

Wilken, M. & Colombo, A.W. 2012, 'Benchmarking Cluster- and Cloud-Computing deciding to outsource or not data processing in industrial applications', Industrial Electronics (ISIE), 2012 IEEE International Symposium on, pp. 1184-90.

Wittenberg, R. 2000, 'Abbas Tashakkori und Charles Teddlie: Mixed methodology. Combining qualitative and quantitative approaches. Applied social research method series, volume 46', KZfSS Kölner Zeitschrift für Soziologie und Sozialpsychologie, vol. 52, no. 1, pp. 180-3.

Xingchen, L., Weimin, L. & Wei, Z. 2012, 'The Design and Implementation of XMPP-Based SMS Gateway', Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on, pp. 145-8.

Xu, G., Pang, J. & Fu, X. 2013, 'A load balancing model based on Cloud partitioning for the public Cloud', Tsinghua Science and Technology, vol. 18, no. 1, pp. 34-9.

Yamamoto, H., Maruta, D. & Oie, Y. 2005, 'Replication methods for load balancing on distributed storages in P2P networks', Applications and the Internet, 2005. Proceedings. The 2005 Symposium on, pp. 264-71.

Yanmei, H., Hongyuan, W., Liang, H. & Yang, H. 2011, 'A Cloud Storage Architecture Model for Data-Intensive Applications', Computer and Management (CAMAN), 2011 International Conference on, pp. 1-4.

Yang, W., Bin, Z., Ying, L. & Deshuai, W. 2010, 'The modeling tool of SaaS software', Advanced Computer Control (ICACC), 2010 2nd International Conference on, vol. 2, pp. 298-302.

Y.-K.Kwok and I.Ahmad, "Static scheduling algorithms for allocating directed taskgraphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406–471,1999.

Yihong, G., Huadong, M., Haitao, Z., Xiangqi, K. & Wangyang, W. 2013, 'Concurrency Optimised Task Scheduling for Workflows in Cloud', *Cloud Computing (CLOUD)*, 2013 IEEE Sixth International Conference on, pp. 709-16.

Yike, G., et al. (2012). Does the Cloud need new algorithms? An introduction to elastic algorithms. *Cloud Computing Technology and Science (CloudCom)*, 2012 IEEE 4th International Conference on.

Yilin, L., Jian, Z., Shaochun, W. & Shujuan, Z. 2012, 'A Hybrid Dynamic Load Balancing Approach for Cloud Storage', *Industrial Control and Electronics Engineering (ICICEE)*, 2012 International Conference on, pp. 1332-5.

Yun, Y., Ke, L., Jinjun, C., Xiao, L., Dong, Y. & Hai, J. An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows. *eScience*, 2008. *eScience '08*. IEEE Fourth International Conference on, 7-12 Dec. 2008 2008. 37-375.

Zenon Chaczko and Germano Resconi. Morphotronic system applications. In Roberto Moreno-Diaz, Franz Pichler, and Alexis Quesada-Arencia, editors, *Eurocast 2009 12th International Conference on Computer Aided Systems Theory*, pages 905\_912. Springer-Verlag, 2009.

Zhang, Q., Cheng, L. & Boutaba, R. 2010, 'Cloud Computing: state-of-the-art and research challenges', *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18.

Zhangjun, W., Zhiwei, N., Lichuan, G. & Xiao, L. A Revised Discrete Particle Swarm Optimisation for Cloud Workflow Scheduling. *Computational Intelligence and Security (CIS)*, 2010 International Conference on, 11-14 Dec. 2010 2010. 184-188.

Zhan-sheng, L., Da-wei, L. & Hui-juan, B. 2008, 'CRFP: A Novel Adaptive Replacement Policy Combined the LRU and LFU Policies', *Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on*, pp. 72-9.

Zhen, X., Guo, L. & Tracey, J. 2007, 'Understanding Instant Messaging Traffic Characteristics', Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on, pp. 51-.

Zheng, L., O'Brien, L., He, Z. & Cai, R. 2012, 'On a Catalogue of Metrics for Evaluating Commercial Cloud Services', Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, pp. 164-73.

Zhenzhong, Z., Limin, X., Yuan, T., Ji, T., Shouxin, W. & Hua, L. 2013, 'A Model Based Load-Balancing Method in IaaS Cloud', Parallel Processing (ICPP), 2013 42nd International Conference on, pp. 808-16.

Zuo, B. & Lin, S. 2010, 'Multiple QoS-based Best Efficiency Workflow Scheduling', Computer Application and System Modeling (ICCASM), 2010 International Conference on, vol. 15, pp. V15-455-V15-9.

# Chapter 9

## Appendix

### 9.1 Experiment Specifications

#### 9.1.1 HDM System

HDM objective is to optimise the data retrieval from different database in Cloud environments. Based on the users' access permissions, every doctors and nurse can retrieve patients' data including x-ray scans, audios and videos recording quicker and with less waiting time.

Figure 9.1 depicts the overall view of the HDM system that defines the HDM as a separate entity connected to the same or other different Cloud systems. The user interface will connect the users through this complex.

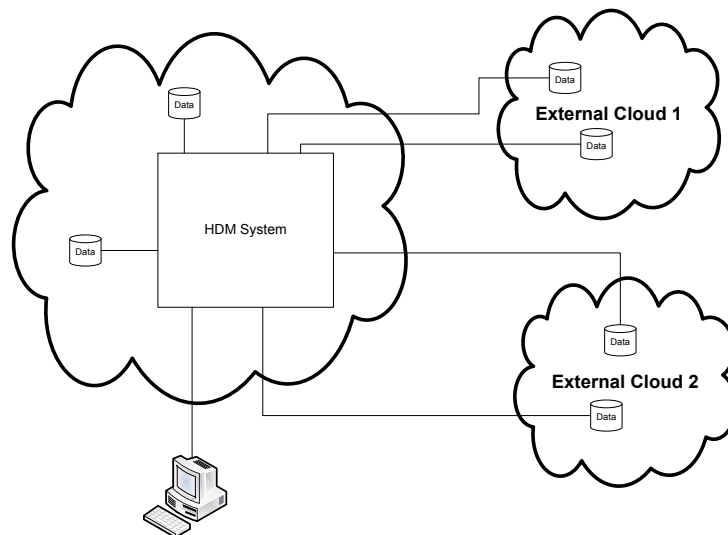


Figure 9.1-HDM system high-level design

The HDM system can scale the resources whenever required by evaluating the usage patterns identified from the past data usage.

The following items are considered in HDM scope:

- Analyse the usage patterns such as CPU usage for forecasting the future needs.
- Anticipate the scalability feature whenever there is a need for that.
- Compress and un-compress data for minimising the costs.

As the HDM Cloud-based system is functioning globally, it will retrieve patients data from all over the world from different sites located on other Clouds. Therefore, proactive scaling and comparison are the main important features of the HDM system.

Proactive scaling provides users with faster data access by forecasting the resources needed based on users' past data access. Compression on the other hands, speed up the transfer rate by encrypting the data. However, it needs more memory for comparison purposes before data transfer could happen.

### **9.1.1.1 Development specifications**

Java programming language has been applied for coding HDM system. The infrastructure of the system is built on open source Cloud environments where two virtual machines are selected to represent the Cloud nodes.

Each virtual machine can be in the high usage or low usage. The database also includes some random files such as images, audios and videos. LZMA algorithm has been coded to replicate the compression functionality needed in this system.

Below describes the requirements gathered for programming the HDM applications:

- **General Requirements**

**SRS-GR01** HDM will be hosted on Cloud infrastructure.

**SRS-GR02** HDM will simulate data usage for Hospital applications.

**SRS-GR03** Nurse and doctors are able to retrieve patients' data such as x-ray, recorded audios and videos.

- **Usage Patterns**

**SRS-UP01** HDM is able to analyse the past users' usage patterns and predict the future needs.

**SRS-UP02** HDM system should consider the future patients appointments for database record.

- **Compression**

**SRS-CO01** HDM will compress the required data before transfer could happen.

**SRS-CO02** HDM is able to un-compress the data upon receiving patients' files.

- **Databases**

**SRS-DB01** The HDM databases is able to store different data formats such as documents, images, audio and videos.

**SRS-DB02** The HDM should store the users' usage patterns.

**SRS-DB03** The HDM database should be able to connect to different Cloud-based systems.

- **Proactive Scaling**

**SRS-PS01** The HDM system shall forecast the future scaling needs.

- **Reporting**

**SRS-RP01** HDM system should have the reporting functionality for depicting the current and future usage patterns.

- **Configuration**

**SRS-CF03** HDM system will define the period for refreshing the usage forecast.

**SRS-CF06** HDM configuration should have functionality to allow switching back to proactive scaling.

## 9.1.2 STEM-PSO Implementation Details

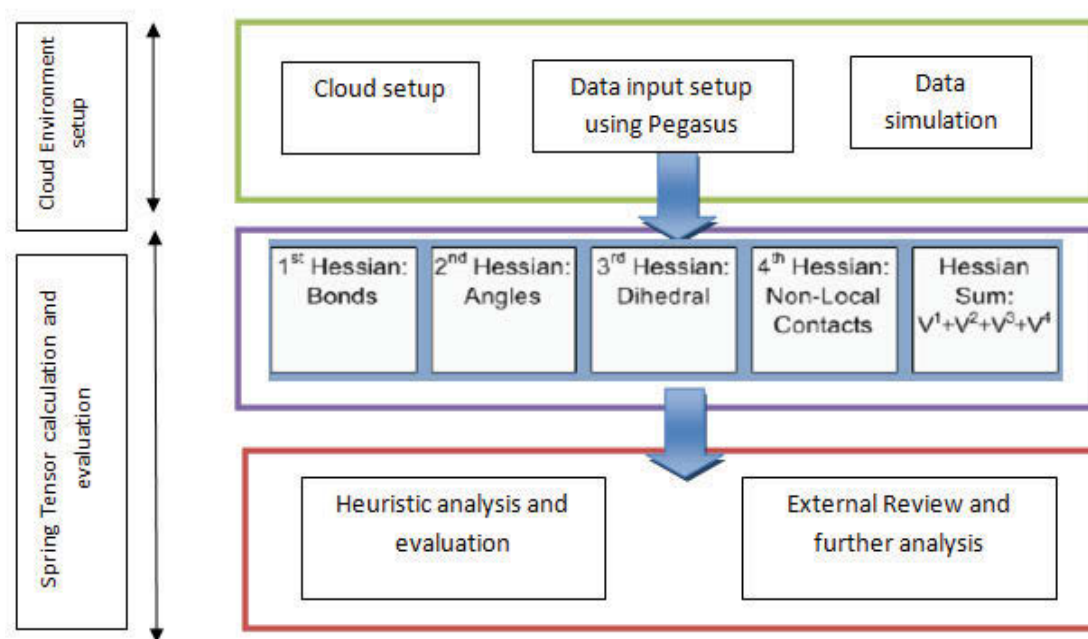


Figure 9.2- STEM-PSO high-level logical design

Figure.9.2 provides the high-level logical concept of STEM-PSO experiment. The first layer depicts the Cloud environment setup while the second layer points to the spring tensor projection functionalities. The last layer performs the calculation and evaluations analysis.

The experiment has been implemented using Java programming language. Moreover, the Pegasus software has been used to generate the data inputs for the experiment. Pegasus is a UNIX platform which can generate excel and XML files. Figure.9.3 is showing a sample code for generating the workflow data input by using Pegasus software.

The STEM-PSO heuristic screenshots are shown in Figure.9.4, Figure 9.5 and Figure 9.6.

Figure 9.7 also, depicts the STEM main class code.



```

remote_tile_setup = Job(namespace="gp", name="remote_tile_setup", version="1.0")
remote_tile_setup.addArguments("%05d" % (tile_id))
remote_tile_setup.addProfile(Profile("dagman", "CATEGORY", "remote_tile_setup"))
remote_tile_setup.uses(params, link=Link.INPUT, register=False)
remote_tile_setup.uses(mdagtar, link=Link.OUTPUT, register=False, transfer=True)
uber dax.addJob(remote_tile_setup)

...

subwf = DAX("%05d.dax" % (tile_id), "ID%05d" % (tile_id))
subwf.addArguments("-Dpegasus.schema.dax=%s/etc/dax-2.1.xsd" % (os.environ["PEGASUS_HOME"]),
                  "-Dpegasus.catalog.replica.file=%s/ro.data" % (tile_work_dir),
                  "-Dpegasus.catalog.site.file=%s/sites.xml" % (work_dir),
                  "-Dpegasus.transfer.links=true",
                  "--sites", cluster_name,
                  "--cluster", "horizontal",
                  "--basename", "tile-%05d" % (tile_id),
                  "--force",
                  "--output", output_name)
subwf.addProfile(Profile("dagman", "CATEGORY", "subworkflow"))
subwf.uses(subdax_file, link=Link.INPUT, register=False)
uber dax.addDAX(subwf)

```

Figure 9.3-Pegasus workflow data generator

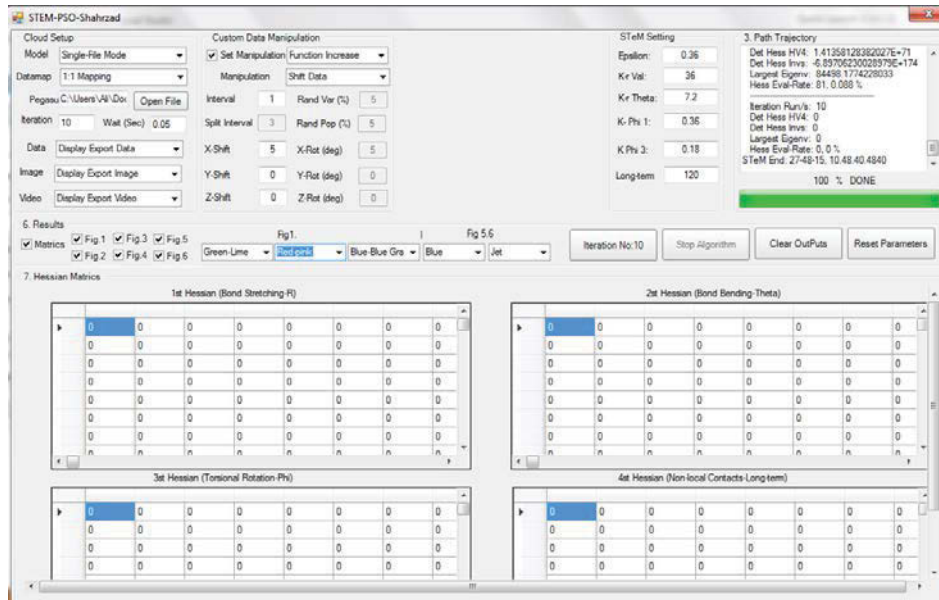


Figure 9.4- STEM-PSO front end

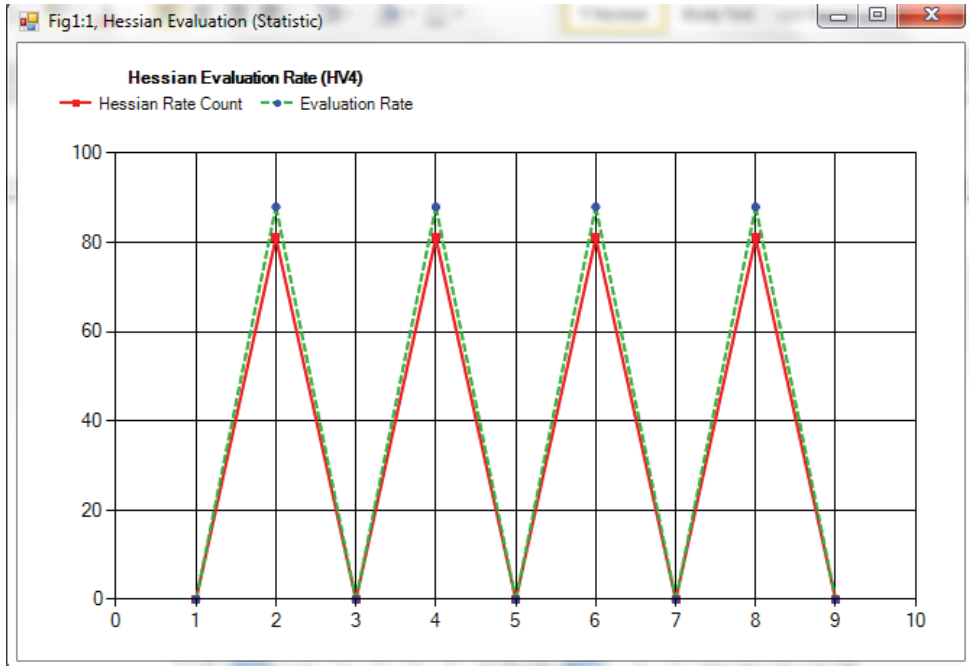


Figure 9.5- Hessian evaluation rate

The dialog box 'PSO Algorithm Input Parameters' contains the following settings:

- PSO Maximun Iteration: (Overrides Iteration Value Set in Main Panel.)
- PSO Teration Factor (Suggested: 1.0)
- PSO Correction Factor (Suggested: 2.0)
- PSO Initial Best Value (Suggested: 1000)
- Goal to Reach (X-Coordinate)
- Goal to Reach (Y-Coordinate)
- Goal to Reach (Z-Coordinate)
- Percentage Adjustment: (Specify % Value from 0 to 100)
- Swam Sienration (If File Input is Used, this parameter will be ignored.)
- Evaluation Function: (Default:  $(X-XGoal)^2 + (Y-YGoal)^2 + (Z-ZGoal)^2$ )

Buttons: Ok

Figure 9.6- PSO algorithm input -front end

```

class STEM
{
    public double[] pdistNew(double[,] dataNew){
        int ElementCount=(dataNew.GetLength(0)*(dataNew.GetLength(0)-1))/2;
        double[] dist=new double[ElementCount];
        int counter=0;
        for (int i=0;i<dataNew.GetLength(0);i++){
            for (int j=i+1;j<dataNew.GetLength(0);j++){
                double sum=0.0;
                for(int k=0;k<dataNew.GetLength(1);k++){
                    sum += ((dataNew[i, k] - dataNew[j, k]) * (dataNew[i, k] - dataNew[j, k]));
                }
                dist[counter]=Math.Sqrt(sum);
                counter++;
            }
        }
        return dist;
    }
}

private double[,] squareformNew(double[] x){

    int rows= x.Length-2;
    int m = (int)((1+Math.Sqrt(1+8*rows))/2);
    double[,] r=new double[m+2,m+2];
    for (int i=0;i<m+2;i++)
        for(int j=0;j<m+2;j++){
            r[i,j]=0;
        }
    int start = 1;
    for(int k = 1;k<=m-1;k++){
        int counterk=k+1;
        for (int start1=start;start1<=start+m-k-1;start1++){
            r[counterk,k] = x[start1];
            r[k,counterk] = x[start1];
        }
    }
}

```

Figure 9.7- STEM-PSO main class code

## 9.1.3 Cloudsim Specification Details

Figure 9.8 depicts the Cloudsim high-level architecture overview that has been used in the second experiment of this thesis.

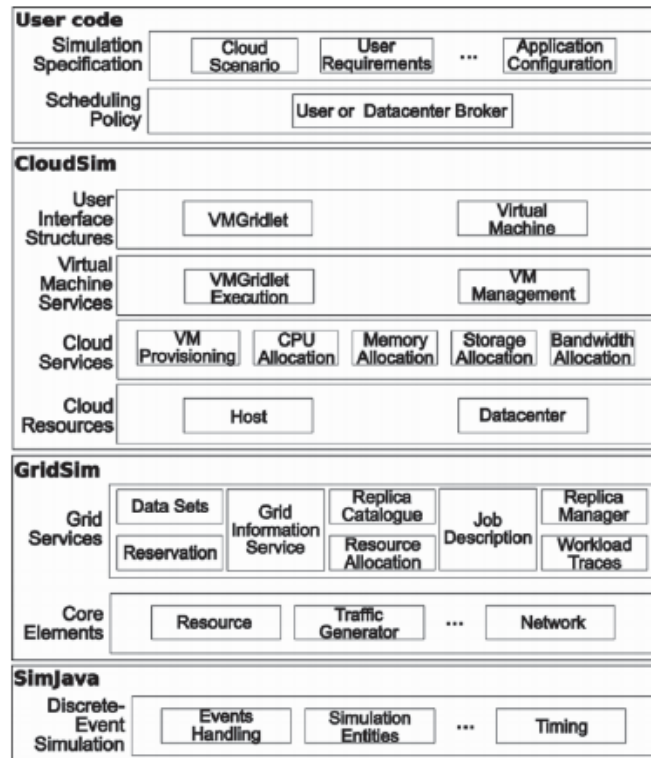


Figure 9.8- Cloudsim high-level design

Cloudsim simulator is developed based on Grid simulation tools. In Grid toolkits, there is no opportunity for on-demand virtualisation for resource management; therefore it cannot run on Cloud infrastructures.

Cloudsim instead can be considered as an upgraded version of Grid simulation tool with virtualisation functionality so it can be implemented on Cloud-based systems.

As it is shown in figure.9.9 Cloudsim is composed of the following main classes:

- Data-centre: is providing the required infrastructures needed in the Cloud platform.
- Broker: is acting as the interface between user's requirements and Cloud services.
- Cloudlets: are representing the tasks submitted as user's requirements.
- VM provisioned: This is an abstract class which represents the provisioning policies for allocating the relevant VMs on Hosts.

Cloudsim is developed using Java language that enables high-level programming services such as queuing and processing the events, creating different components such as hosts, data centres, brokers and virtual machines.

The core component of the simulator architecture is called "Cloudsim" which has all the core functionalities similar to Gridsim libraries.

In this layer, Clouds is managing the VMs, memories and storage needed for Cloud simulation. The top layer of the simulation helps users to setup the configuration and related functionalities such as managing the hosts, managing VMs and updating the scheduling policies. Also, this layer helps developers to conduct a robust test on their customised configuration platform to ensure the efficiency of their task scheduling behaviour.

Next section illustrates the Cloudsim database components, shown in Figure 9.9



Host: The class explains the allocation policies for allocating the core processing power on VMs. It provides the provisioning policy for assigning the memory and bandwidth to VMs. Also, the class contains the detailed information for the available memory and storage.

Network Topology: The network topology contains the information regarding the network behaviour.

Ram Provisioner: The class allocates the required memory on VMs.

VM: This class is also known as Cloudlet scheduler. The class contains information about accessible memory, processor, and storage size.

VM allocation policy: The class is responsible for allocating the VMs on hosts. It is responsible for analysing the status of the hosts from memory and storage capacity for VM allocation.

VM-scheduler: In this class, core processor will be allocated on VMs. The allocation should happen based on the specific allocation policies.

Considering the architecture of the classes in Cloudsim tool, in our experiment we have applied the replication methodology to optimise the load balancing by minimising the finishing time. Figure 9.10 is depicting the Clouds submission time simulation front-end.

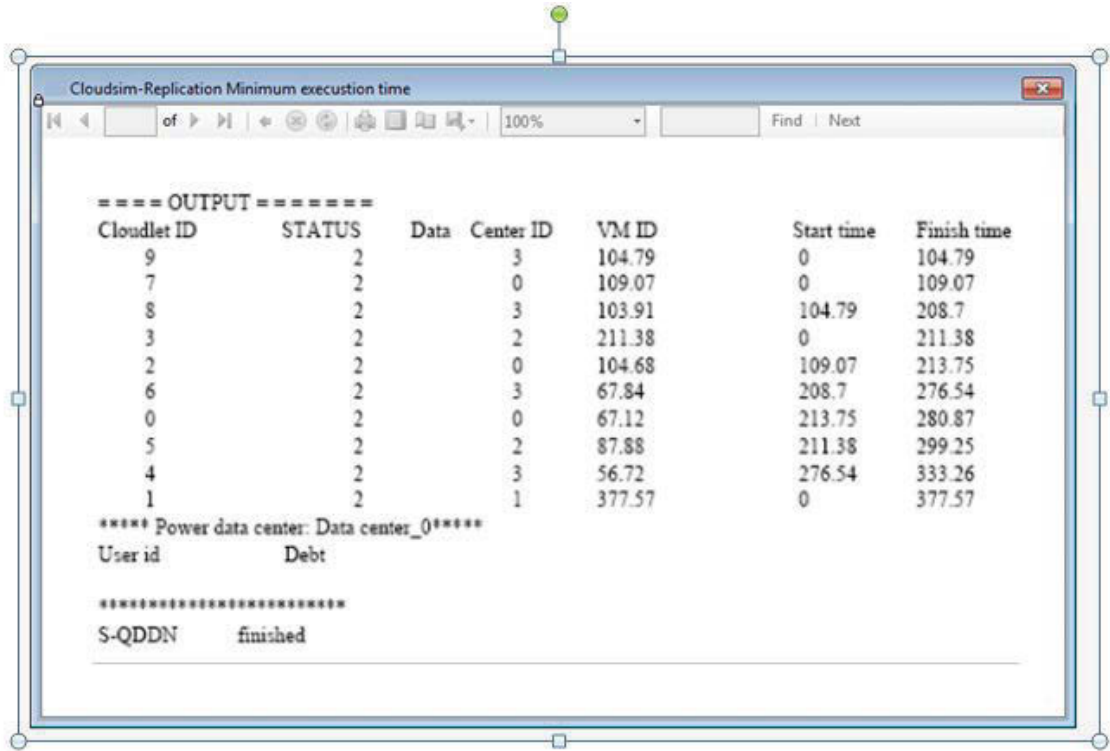


Figure 9.10-Cloudsim submission time frontend



### 9.1.3.1 Adding a new file in Tier structure

Example: adding file (C, b, 100)

The request has been submitted by site C to add file “b” with the access time of 100. Figure 9.11 is depicting the first structure of the tier.

To start, the last pointer of site C will be retrieved. Then the difference between the last time the pointer is pointing to and the time that the new file is accessed will be calculated. If  $t_{submitted\ file} - t_{pointer} > threshold$  then it means that the file should be added as a new sequence otherwise, the new file is considered as a successive file and it will be added as a child where pointer was pointing to.

So in this example pointer is pointing to (z,4,4). As  $100-4 > threshold$  (threshold is this algorithm is 40), the new branch would be added to the site.

In this step, the third tier will be searched to find the site that has file “b”. If the file wasn’t in this tier the search will continue through next tier. As the figure shows file b is located on tier 3, so the access time will be updated to 100 and the number of access will be incremented. Pointer in site C now will point to this node, Figure 9.12.

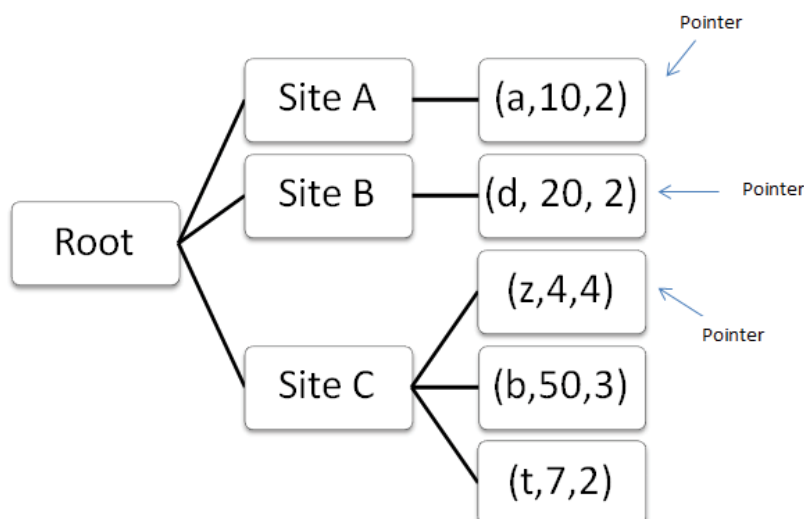


Figure 9.11- Inserting a node in tier structure

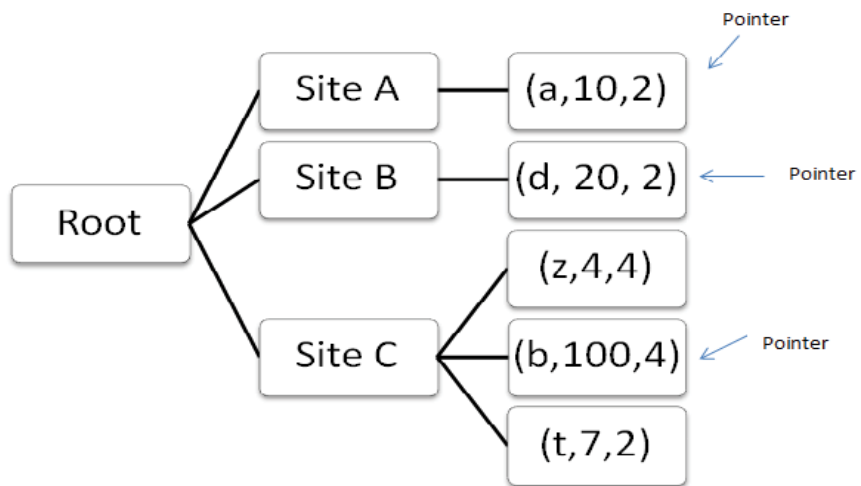


Figure 9.12- Updating the pointer place in tier structure