Energy-Aware Scheduling in Decentralised Multi-Cloud Systems

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

By

Aeshah Yahya Alsughayyir

Department of Informatics
University of Leicester

April 2018

Energy-Aware Scheduling in Decentralised Multi-Cloud Systems

Aeshah Yahya Alsughayyir

Abstract

Cloud computing is an emerging Internet-based computing paradigm that aims to provide many on-demand services, requested nowadays by almost all online users. Although it greatly utilises virtualised environments for applications to be executed efficiently in low-cost hosting, it has turned energy wasting and overconsumption issues into major concerns. Many studies have projected that the energy consumption of cloud data-centres would grow significantly to reach 35% of the total energy consumed worldwide, threatening to further boost the world's energy crisis. Moreover, cloud infrastructure is built on a great amount of server equipment, including high performance computing (HPC), and the servers are naturally prone to failures.

In this thesis, we study practically as well as theoretically the feasibility of optimising energy consumption in multi-cloud systems. We explore a deadline-based scheduling problem of executing HPC-applications by a heterogeneous set of clouds that are geographically distributed worldwide. We assume that these clouds participate in a federated approach. The practical part of the thesis has focused on combining two energy dimensions while scheduling HPC-applications (i.e., energy consumed for execution and data transmission). It has considered simultaneously minimising application rejections and deadline violations, to support resource reliability, with energy optimisation. In the theoretical part, we have presented the first online algorithms for the non-preemptive scheduling of jobs with agreeable deadlines on heterogeneous parallel processors.

Through our developed simulation and experimental analysis using real parallel workloads from large-scale systems, the results evidenced that it is possible to reduce a considerable amount of energy while carefully scheduling cloud applications over a multi-cloud system. We have shown that our practical approaches provide promising energy savings with acceptable level of resource reliability. We believe that our scheduling approaches have particular importance in relation with the main aim of green cloud computing for the necessity of increasing energy efficiency.

Acknowledgements

In the name of Allah, the most merciful, the most beneficent.
Above all, I am thankful to Allah Almighty for his blessings
and for giving me the capability to accomplish this work.

I firmly confess that this section is the most one I found hard to write, and I have kept rewritten it many times; all this because of my desire to heartily express my feelings towards some memorable people. But I ultimately concluded that the gratitude I feel to them cannot be expressed in any language. Eventually, moving all things from our universally psychological language to any specific spoken/written-language is certainly impossible, nonetheless, below is my best attempt.

One of Allah Almighty great blessings on me was supervision of Professor Thomas Erlebach, who gave me the freedom to explore the world of academic research and guided me, in an interesting manner, throughout my degree. It was him who kept me on a thoughtful and well-planned track, and always there for help when I got lost with my own thoughts. Written words are absolutely not enough to acknowledge you Thomas to what you have done not only for me but also for all students you have had supervised. I give you heartfelt thanks and sincerely wish you a wonderful life ahead with no tensions or worries.

I would like to thank my second supervisor Prof. Rajeev Raman, and the annual committee, Dr. Stephan Reiff-Marganiec and Dr. Fer-Jan de Vries, for their great discussions, comments, and kind help in several occasions. I also thank the external examiner Dr. Ligang He for the valuable feedback and comments.

I am deeply grateful to my parents and grandmother, who have provided me through moral and emotional support in my life. To my mother: I love you, and no words are enough to express my heartfelt thanks for your continuous love, prayers, and for looking after my daughter sometimes whilst I was studying. To

my parents-in-law and siblings: I cannot thank you enough for your encouragement and prayers throughout my research. To my husband (Dr. Abdullah) and my beautiful daughter (Abrar): I am profoundly indebted to you for your love, help, support, patience, and understanding.

In the distant past of my life, getting into this academical research adventure was something of a fantasy, and it would not have been possible without the generous support from my sponsor, Taibah University in Madinah. They have offered me a fully funded scholarship, which covered all expenses for me and my family throughout my studies. I sincerely give my profound gratitude to you 'Taibah University', and promise to do everything I could to return the favor. In the same context and support, I extend my sincerest appreciation to the Department of Informatics of the University of Leicester for giving me a great study environment, and also for supporting me to attend several international conferences.

Aeshah Alsughayyir

Table of Contents

Abstract i

Acknowledgements ii

List of Tables vii

List of Figures viii

Abbreviations x

1 Introduction 1

1.1 Cloud Computing and Energy Concern 1

1.1.1 Utilisation of Cloud Resources 2

1.1.2 Multi-Cloud Computing and Cloud Federation 3

1.2 Scheduling Approach for Energy Optimisation: Problem and Objectives 5

1.3 Thesis Contributions and Publications 7

1.4 Thesis Structure 10

2 Relevant Background and Related Work 12

2.1 Cloud Computing Infrastructure 13

2.2 Energy Efficiency in the Cloud System 15

2.2.1 Energy-Saving Techniques 17

2.2.2 Data Computing and Transmission Energy 19

2.3 Scheduling Concepts 20

2.3.1 Definitions 21

2.3.2 Robust Scheduling Under Uncertainty 22

2.3.3 Scheduling Complexity 23

2.3.4 Online Speed-Scaling and Competitiveness 24

2.4 Related Work and Comparison to the State of the Art 25

2.4.1 Scheduling in Decentralised Systems 28

2.4.2 Green Scheduling in Multi-Cloud System 32

2.4.3 Performance Evaluation in Cloud Computing 35

2.4.4 Speed-Scaling on Parallel Processors 37

2.5 Summary 40

3 A Simulation-Based Framework for Analysis and Evaluation 42
3.1 Proposed Models 43
3.1.1 Multi-Cloud Model 43
3.1.2 HPC-Application Model 45
3.2 MCST: Prototype of a Multi-Cloud Simulation Tool 46
3.2.1 Structural Model and Implementation 48
3.2.2 Simulation Stages and Execution Flows 51
3.3 Comparison to Existing Tools 55
3.4 Summary 58

4 Energy Optimisation with Best-Effort Scheduling Mode 60
4.1 Scheduling and Energy Models 61
4.1.1 Scheduling Framework 62
4.1.2 Energy Formula for Application Execution 63
4.1.3 Energy Formula for Data Transmission 67
4.1.4 Problem Formulation 68
4.2 Schedule Decision Strategies 69
4.2.1 Preference Rate Strategy (PRS) 70
4.2.2 Combination Rate Strategy (CRS) 71
4.3 Scheduling Algorithms 74
4.3.1 EGSBE: Energy-Aware Global Scheduling Based on the Best-Effort Mode 76
4.3.2 ELS: Energy-Aware Local Scheduling Algorithm 78
4.4 The Evaluation of the Proposed Schedulers 79
4.4.1 Configurations and Input Applications 80
4.4.2 Effect of the Proposed Schedulers on Energy Optimisation 82
4.4.3 Impact of the Proposed Scheduling Strategies on Resource Utilisation 89
4.4.4 Discussion and Experiment Findings 90
4.5 Summary 92

5 Energy Optimisation with Advance-Reservation Mode 94
5.1 Definition and Illustration of Scheduling Problem 96
5.2 EGSAR: Energy-Aware Global Scheduling with Advance Reservation 99
5.3 Experimental Evaluation 103
5.3.1 Configurations 104
5.3.2 Effect of the Proposed Schedulers on Energy Optimisation 106
5.3.3 Impact of the Resource Failures on Scheduled Applications 109
5.3.4 Discussion and Experiment Findings 110

5.4 Comparison with the Best-Effort Approach 113
5.5 Summary 117

6 Non-Preemptive Scheduling on Power-Heterogeneous Processors 120
6.1 Problem Definition 122
6.2 Non-Preemptive AVR 123
6.3 Jobs with Small Density Ratio and Interval-Length Ratio 125
6.3.1 Jobs with Equal Release Time, Deadline, and Density . . 125
6.3.2 Jobs with Equal Interval Length and Density 128
6.3.3 Interval Lengths and Densities Within a Factor of Two . 131
6.4 Arbitrary Interval Lengths and Densities 135
6.5 Summary 138

7 Conclusion and Future Work 139
7.1 Summary of Contributions and Achievements 142
7.2 Limitations and Assumptions 144
7.3 Future Work 145
7.3.1 Extension of the Considered System Model 145
7.3.2 Elaborating and Enriching our Scheduling Approaches . 147

Bibliography 149

List of Tables

2.1 Pros and cons of centralised and decentralised computing systems [78]. 14
2.2 A summarised comparison between the relevant approaches to this thesis 28
2.3 Known and new (in bold) results for speed-scaling on parallel processors. S stands for homogeneous and S^* for heterogeneous processors 40

3.1 Cloud parameters of a homogeneous data-centre. 44
3.2 The main configurations and options for managing the process of MCST, defined in the class *IUScheduler* 51
3.3 Comparison between the closely related simulation tools with our MCST 57

4.1 An example describing how our proposed combination rate strategy (CRS) can be applied. 73
4.2 Specification of the five clouds used in our simulation. 81
4.3 Approximate distances in km between the cloud data-centres. . 81
4.4 Three categories of parallel workload applications. 81
4.5 Details of the energy usage and other values obtained from scheduling three different workloads vs. different strategies, based on the best-effort mode. 86

5.1 Details of the energy usage and other values obtained from scheduling three different workloads vs. different strategies, based on the advance-reservation mode. 106
5.2 Comparison between the best-effort and the advance-reservation scheduling modes in the number of rejected and violated applications based on the preference and combination rate strategies. 115

List of Figures

3.1 Proposed model representing distributed heterogeneous multi-cloud structure, which includes two scheduler components, discussed in detail in the mentioned chapters 44
3.2 An example of a simple DAG application, consisting of 7 tasks . 45
3.3 Class diagram of the MCST main classes 49
3.4 Task's states and defined transitions in our application model . 49
3.5 An example of a statistical report generated by our tool (MCST). It shows the details after scheduling and executing 50 HPC-applications over 5 distributed clouds. Similar generated reports have been made publicly available 54
3.6 The sequential extensions between the related simulation tools to our MCST, that are inherited originally from SimJava [63] . . 56

4.1 Overview of the proposed best-effort approach, described in three steps 64
4.2 Energy consumption vs. frequency 66
4.3 An illustrative example explaining abstractly the form of the clouds' offers. It represents some assumed values, in two lists, offered by 5 clouds. 70
4.4 Applying the CRS strategy to the example presented in Figure 4.3, which results in the choice of *Cloud D* as the best option. 73
4.5 Different scheduling strategies with various workload categories vs. the objective function. 83
4.6 Different scheduling strategies with various workload categories vs. the rate of energy usage. 84
4.7 Different scheduling strategies with various workload categories vs. the number of failed applications. 85
4.8 An estimation, based on the results presented in Figure 4.6, to break down the overall reduction of the consumed energy into the two considered energy saving methods: DVFS and sharing globally the available cloud's resources. The table above also summarises the comparison between the scheduling strategies in terms of the energy reduction according to the objective function. 87

4.9 Different scheduling strategies with 50 submitted applications from the mid-load category vs. the extension of application deadlines in increments of 20%. 88
4.10 Resource utilisation with preference rate Strategy PRS. 90
4.11 Resource utilisation with combination rate strategy CRS. 91

5.1 Overview of the proposed advance-reservation approach, described in three steps. 97
5.2 Different scheduling strategies with various workload categories vs. the objective function. 105
5.3 Different scheduling strategies with various workload categories vs. the rate of energy usage. The deadlines of executing all these applications have been extended based on loose deadlines. . . . 107
5.4 Different scheduling strategies with various workload categories vs. the number of rejected applications. The deadlines of executing all these applications have been extended based on loose deadlines. 108
5.5 Different scheduling strategies with 50 submitted applications from the mid-load category vs. the extension of application deadlines in increments of 20%. 109
5.6 Number of violations vs. percentage of down servers. 110
5.7 Resource utilisation with preference rate Strategy PRS. 112
5.8 Resource utilisation with combination rate strategy CRS. 112
5.9 Comparison between the best-effort and the advance-reservation scheduling modes in the rate of energy usage based on the preference and combination rate strategies. 115
5.10 Comparison between the best-effort and the advance-reservation scheduling modes in the network utilisation, represented by the number of messages that are generated by MCST as SimJava events. 116
5.11 Comparison between the best-effort and the advance-reservation scheduling modes in the percentage of energy reduction based on the preference and combination rate strategies. 117

6.1 AVR and NAVR schedules for an example with 3 jobs 124

Abbreviations

ALG	The energy cost of some given **ALG**orithm.
AR	**A**dvance **R**eservation.
AVR	**AV**erage **R**ate algorithm.
BE	**B**est **E**ffort.
CMMS	**C**loud **M**in-**M**in **S**cheduling algorithm.
CRS	**C**ombination **R**ate **S**trategy.
CRS.HF	**C**ombination **R**ate **S**trategy with **H**ighest **F**requency mode.
D	**D**ensity ratio.
DAG	**D**irected **A**cyclic **G**raph.
DEM	**D**ynamic **E**nergy **M**anagement.
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling.
EAT	**E**arliest **A**vailable **T**ime.
EC2	**E**lastic **C**loud **C**omputing.
ELS	**E**nergy-aware **L**ocal **S**cheduling algorithm.
EST	**E**arliest **S**tart **T**ime.
GESAR	**G**lobal **E**nergy-aware **S**cheduling with **A**dvance **R**eservation algorithm.
GESBE	**G**lobal **E**nergy-aware **S**cheduling with **B**est **E**ffort algorithm.
HPC	**H**igh **P**erformance **C**omputing.
IaaS	**I**nfrastructure **as a** **S**ervice.
InPs	**In**frastructure **P**rovider**s**.
IT	**I**nformation **T**echnology.
LFT	**L**atest **F**inish **T**ime.
MCST	**M**ulti-**C**loud **S**imulation **T**ool.

NAVR	**N**on-preemptive **AV**erage **R**ate algorithm.
OPT	The energy cost of an **OPT**imal schedule.
P	**P**rocessor.
P&P	**P**ush **and** **P**ull model.
PR	**P**rovisional **R**eservation.
PRS	**P**reference **R**ate **S**trategy.
PRS.HF	**P**reference **R**ate **S**trategy with **H**ighest **F**requency mode.
RSD	**R**ate of **S**tandard **D**eviation.
SD	**S**tandard **D**eviation.
SLA	**S**ervice **L**evel **A**greement.
T	**T**ime interval-length ratio.
VM	**V**irtual **M**achine.

Chapter 1

Introduction

1.1 Cloud Computing and Energy Concern

A cloud computing[1] system, in simple terms, is an Internet-based computing system that aims to provide many on-demand services, requested nowadays by almost all online customers. Rather than constructing a costly data-centre, equipped with many servers and storage devices, etc., for processing and managing applications, cloud computing can handle not only application deployment and hosting but also required devices (particularly, high performance computing (HPC) servers) to be delivered through the Internet as services. Such overwhelming services and supports are obviously derived from a magnificent and powerful infrastructure. Typically, cloud infrastructure is built upon a large number of servers, including HPC and massive storage devices, that need huge energy supplies.

The advent of cloud computing, although it greatly supports virtualised environments for applications to be executed efficiently in low-cost hosting, has turned energy wasting and overconsumption issues into major concerns. As a worrying

[1]The word *cloud* is a metaphor for the *Internet*, hence, cloud computing refers to Internet-based computing [100]

manner, many studies (some of them are discussed in [90]) have projected that the energy consumption of cloud data-centres would grow significantly to reach 35% of the total energy consumed worldwide [114], threatening to further boost the world's energy crisis [99]. These studies have emphasised the importance of making cloud infrastructures energy efficient, and this has mostly motivated us to carry out this thesis.

A recent study [54] reports that cloud data-centres consume approximately up to 3% of the total energy consumption around the world, and the consumption is projected to grow significantly to 1012.02 billion kWh by 2020 [114]. Another study [49] (cited in [90]) also estimates that a typical data-centre consumes about $27,048$ kWh per day and this amount is equivalent or even more to what 2,500 households would consume in the EU. Such a huge amount of energy is probably due to the notable growing demand for cloud services in many sectors, such as the evolution of eScience, gaming systems, social big data analysis, and data mining applications. Moreover, a figure depicted in [112] estimates that the majority of energy in a typical cloud data-centre is consumed by server computing activities, while less than 50% is needed for all other components such as storage and cooling systems. This has also motivated us to focus on the server component in our proposed solution in general.

In Section 2.1 and Section 2.2, we extend our discussion regarding cloud computing infrastructures and their energy issues. However, in the next sections, we will narrow the discussion down, focusing on energy optimisation by efficiently utilising a set of cloud resources, toward explaining our research problems.

1.1.1 Utilisation of Cloud Resources

A cloud computing system depends principally on sharing computing resources from one or more data-centres. This has brought the efficient utilisation of cloud research under the spotlight that raises the necessity of (1) addressing energy overconsumption issues and of (2) supporting the reliability of resource utilisation as cloud resources are naturally prone to failures [103]. We attempt in this thesis to handle these two points.

Concerning the energy overconsumption issue, many pioneering researchers have devoted their studies, surveyed in [73], to improving the utilisation of cloud resources. To a large extent, the core factors behind efficient resource utilisation are the high capabilities of virtualisation techniques [101] as well as the flexibility of dynamically adjusting voltage and frequency of processors [65]. These techniques provide an effective way to save energy as virtualisation enables a reduction of the number of active physical machines (relying on Virtual Machine Migration and Consolidation techniques [73]) and, in turn, sharing bounded resources by virtually creating further machines or CPUs to potentially handle large workloads. Dynamic Voltage and Frequency Scaling (DVFS), however, enhances the energy efficiency of executing a workload that is considered in this thesis, discussed in detail in Section 2.2.1. The idea of DVFS is to reduce the supplied voltage for a processor as much as possible while the desired performance, represented by an execution time bound, is still achievable [65]. In this setting, determining the critical frequency when scheduling a task depends mainly on measuring the status of resource utilisation.

1.1.2 Multi-Cloud Computing and Cloud Federation

Multi-cloud computing has come on the technological scene to further augment and expand the shared pool of cloud resources for even more efficient services. It intuitively follows the distributed computing system that goes beyond the capability of a single cloud system in providing critical services, such as backup and disaster recovery. Here, we devote particular attention to a cloud federation system that fundamentally gathers different cloud providers (with heterogeneous infrastructures) to cooperate for many purposes, such as improving cloud services, increasing productivity, and meeting complex customer demands, etc. [88]. For instance, cloud customers may benefit from the federation of cloud resources so as to have more flexible features for mitigating the economic problem of vendor lock-in, gaining better performance with probably cheaper cost [80].

Cloud federation is still an emerging computation paradigm and many approaches with strategic visions have been suggested, surveyed in [75], such as InterCloud vision [32], Cross-Cloud federation approach [38], Multi-Cloud approach [21], and Federated Cloud Management [89]. Broadly speaking, the federative pool of cloud resources are ordinarily designed from a heterogeneous set of cloud providers, owned by either one vendor (such as amazon web services (AWS) cloud computing [8], and some deployed open source cloud systems with distributed data-centres that support AWS API, e.g., Open-Nebula [93], Eucalyptus [50], CloudStack [42], and Nimbus [91]); or multiple vendors (e.g., Open Cirrus testbed[2] [92]). In this thesis, the considered framework assumes the latter design (i.e., a federated heterogeneous set of clouds owned by different providers) with promoting the following particular goals of cloud federation:

- allowing the execution of workloads across multiple providers; and
- optimising the overall energy consumption federation wide.

In the next section, we explain our intention for optimising energy consumption in multi-cloud systems as the main goal. We will firstly shed light on the problem we want to solve, focusing on scheduling cloud applications, and then detail our research objectives.

[2]Open Cirrus is a federated heterogeneous multi-clouds system that consists of more than 10 cloud sites across the globe [12].

1.2 Scheduling Approach for Energy Optimisation: Problem and Objectives

In line with green cloud computing that recognises the necessity of increasing energy efficiency and minimising global warming as well as air pollution [29], this thesis aims to study practically as well as theoretically the feasibility of optimising energy consumption in multi-cloud systems. It attempts to find appropriate solutions to the problem of scheduling cloud applications, focusing ultimately on minimising the wasted energy and/or overconsumption. The purpose of our practical study is to estimate the amount of energy that one can reduce, depending on several realistic energy parameters, whereas, the theoretical study helps to understand how far the produced energy consumption of a given algorithm is from the optimum.

In particular, we consider a deadline-based scheduling problem of distributing HPC-applications over a decentralised multi-cloud system consisting of cloud data-centres that are geographically located in different places around the world. We assume that these clouds are heterogeneous, belonging to different vendors, and participate in a federated approach for sharing the execution of workloads with the goal of optimising the overall energy consumption worldwide. However, this assumption does not mean that this study is not applicable to non-federated systems, e.g., clouds, grids or any other computing clusters. Our approach can be applied to any decentralised system with different clusters of clouds in different locations where there is an issue of scheduling scientific workflows in an energy-efficient way over the whole system. To be more specific, we explain in some detail the main objectives of this thesis, as follows.

First of all, cloud infrastructures are fundamentally data-centres, which often demand high energy usage to permanently operate [23]. Thus far, many common solutions devoted to efficiently scheduling and/or running HPC-workloads have focused on minimising energy usage at a single cloud, but not across multiple clouds. To bridge this gap, we aim to propose an efficient scheduling approach for optimising energy consumption over a global scale as in multi-cloud computing.

Secondly, to tackle energy-aware application scheduling over geographically distributed clouds (owned by different providers), it is important to pay crucial attention to the energy consumed by dataset transmission. To our knowledge, this has not been addressed in the literature yet. In addition to accounting for the energy consumption from both processing and dataset transmission while performing global scheduling, one faces a natural trade-off between energy minimisation and conflicting objectives such as quality-of-service optimisation. Hence, we try to handle, in our proposal, both energy dimensions (i.e., energy consumed for execution and data transmission) along with combining two conflicting objectives.

As consequences, the major part of this thesis has been devoted to examining our energy optimisation algorithms for scheduling HPC-applications. These algorithms are applied to decentralised cloud systems taking the energy usage of dataset transmissions into account. The optimisation supports the combination of two conflicting objectives, minimising both energy consumption and application deadline violation caused by resource failures. The latter objective reflects an important aspect of the quality that makes the proposed scheduling robust. In a sense, our scheduling algorithms attempt to execute cloud applications in a robust manner within cloud data-centres, such that the state of executing applications would remain normal during unforeseen technical failures of cloud servers.

Although we consider minimising application deadline violation, which is an essential objective in the context of scheduling cloud applications for satisfying Service-Level Agreement (SLA)[3], we should mention that the core parts of this thesis as well as the main contribution fall within the scope of energy optimisation in the first place.

1.3 Thesis Contributions and Publications

The prime problem addressed in this thesis is a feasible optimisation of energy consumption in distributed cloud systems, which is in fact one of the core targets of green cloud computing. Our attempts to tackle this problem have already made us contribute to the field of energy-aware scheduling of cloud applications over multi-cloud systems. The novelty of our contributions as far as this thesis is concerned lies broadly in two proposals. The first proposal presents two technical approaches for practically scheduling HPC-applications that consist of dependent tasks. The second proposal presents theoretically the first online algorithms for the non-preemptive scheduling of tasks with agreeable deadlines on heterogeneous parallel processors. These online algorithms rely on the use of competitive analysis to measure the produced energy cost of the considered scheduling method. In the following list, we describe briefly our main contributions:

- an energy-aware global scheduling algorithm with best-effort (EGSBE) for allocating HPC-applications to participating clouds, based on DVFS and the cost of computational-execution [6] as well as dataset transmission;
- an energy-aware local scheduling algorithm (ELS) for mapping each task to required resources [5].

[3]Service-Level Agreement (SLA) is a commitment contract between parties (e.g., service provider and requester) that often includes aspects such as availability, quality, robustness, responsibilities

- an energy-aware global scheduling algorithm (EGSAR) with advance-reservation for allocating HPC-applications to participating clouds, based on DVFS and the cost of computational-execution as well as dataset transmission [5];
- an interdependent decision-making algorithm (referred to as combination rate strategy) to address conflicting objectives using a statistical approach [5];
- another decision-making algorithm (referred to as preference rate strategy) to optimise energy consumption based on setting an upper limit for the allowed energy consumption [5];
- a developed prototype multi-cloud simulation tool MCST[4] to mimic running HPC-workloads for analysis and evaluation.
- an implementation of an existing scheduling algorithm CMMS [84] for comparison purposes [6, 5].
- theoretical algorithms for non-preemptive scheduling of tasks with deadline constraints on heterogeneous processors [7].

In greater detail, we outline our contributions in three points, as follows.

The best-effort and the advance-reservation scheduling approaches: Proposing and experimentally evaluating these two different approaches for optimising the overall energy consumption, accompanied with the minimisation of application rejection/violation cases, when scheduling HPC-applications over a geographically distributed heterogeneous clouds system. These approaches depend on our proposed local scheduler for scheduling application tasks locally in cloud resources using the dynamic voltage and frequency scaling to reduce

[4]The API documentation of our MCST simulation has been made publicly available at `http://www.cs.le.ac.uk/people/ayya1/MCSTdoc/index.html`

energy consumption. Furthermore, these approaches comprise a suggestion of two strategies for scheduling decisions: *Preference Rate* which needs to be predefined, and *Combination Rate* that works dynamically, aiming to address conflicting objectives using a statistical approach. The novel contribution here is to precisely schedule applications that consist of dependent tasks, based on a combination of resource occupation and two energy dimensions: energy consumed for execution and data transmission. Our practical evaluation includes a comparison with an existing scheduling algorithm CMMS [84] besides a number of experiments conducted to widely assess the outcomes.

The multi cloud simulation tool: Presenting a prototype simulation tool developed to evaluate our best-effort and advance-reservation scheduling approaches. The MCST is entirely implemented in Java, and it provides an environment to abstractly represent distributed multi-cloud systems as well as HPC-applications to analyse the scheduling process of tasks and/or HPC-applications for energy efficiency.

Non-preemptive scheduling on heterogeneous processors: Presenting and analysing a non-preemptive online scheduling problem for allocating independent tasks with deadline constraints to heterogeneous processors. This problem fundamentally represents a non-preemptive version of the classical speed-scaling scheduling problem that was studied by Yao et al. [118] but on parallel processors.

In addition to the contributions discussed previously, it is worth mentioning that the major contents of this thesis have been peer-reviewed and published in three conference papers, as listed below:

- Aeshah Alsughayyir and Thomas Erlebach. "Energy Aware Scheduling of HPC Tasks in Decentralised Cloud Systems". In: *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016, Heraklion, Crete, Greece, February 17-19, 2016.* 2016, pages 617–621

- Aeshah Alsughayyir and Thomas Erlebach. "A Bi-objective Scheduling Approach for Energy Optimisation of Executing and Transmitting HPC Applications in Decentralised Multi-cloud Systems". In: *Proceedings of the 16th International Symposium on Parallel and Distributed Computing, ISPDC 2017, 3-6 July 2017, Innsbruck, Austria. (To appear).* 2017

- Aeshah Alsughayyir and Thomas Erlebach. "Online Algorithms for Non-preemptive Speed Scaling on Power-Heterogeneous Processors". In: *Proceedings of the 11th Annual International Conference on Combinatorial Optimization and Applications, COCOA 2017, December 16-18, 2017, in Shanghai, China. (To appear).* 2017

1.4 Thesis Structure

Following this introductory chapter, an overview of the essential concepts and background related to the most important aspects of the thesis are explored in Chapter 2, including a review of the main relevant work. The practical part of this thesis is then introduced in three chapters as follow:

- Chapter 3 presents firstly the considered system model that covers the structure of our multi-cloud and HPC-application models. Next, it focuses on explaining the design and implementation of the proposed prototype simulation MCST that depends principally on the discrete-event method. The chapter will then give a discussion on the most relevant existing software simulations, explaining how our tool differs from them.

- Chapter 4 proposes the best-effort scheduling method (i.e., non-advance-reservation style) for optimising the energy consumption when executing dependent HPC tasks by a decentralised multi-cloud system. In conjunction with energy optimisation as the main objective of this thesis, the proposed scheduling method supports maximising the resource reliability by the reduction of rejected/violated application cases. A preliminary version of this scheduling method has been published in [6], which concentrates on only optimising the computing-energy consumption.

- Chapter 5 proposes a comparable scheduling method with the best-effort for the same objectives. However, this scheduling method follows the advance-reservation style (i.e., needs to ensure that the time frame of an application is scheduled within the resource capacity, taking into account already occupied and reserved resources) when allocating HPC-applications to clouds for execution. The core parts of this chapter, including most of the conducted experiments, have been published in [5].

In Chapter 6, we investigate theoretically a simplified form of our scheduling problem, relying on competitive analysis. It proposes a non-preemptive scheduling approach of jobs with deadline constraints on heterogeneous processors, published in [7]. In conclusion, Chapter 7 summarises our contributions, details the limitations of the proposed approach, and finally explores some prospective outlooks for improving/extending our work.

Chapter 2

Relevant Background and Related Work

Many different practical as well as theoretical approaches have been suggested to address energy-aware scheduling problems that arise in distributed cloud systems or at the parallel processors level. In line with them, this thesis particularly contributes to the development of federated multi-cloud systems for energy efficiency. To clarify the novelty of our contributions, which include a technical method that has been experimentally examined using a simulation tool and also a theoretical based approach, this chapter begins with giving an overview of the essential concepts and definitions related to three important aspects of this thesis, organised in three sections as follows.

The first section provides a brief discussion of cloud computing infrastructure and its types in a wider context. The second section focuses on defining energy efficiency within the common cloud systems. It includes a discussion on the main power components in the cloud environment, and more importantly, it sheds light on the existing energy saving techniques. In the third section, some basic knowledge about the scheduling problem including some related concepts at an abstract level of the problem are reviewed in terms of the complexity theory perspective. Following that, this chapter further discusses the most relevant work in terms of the decentralised environment and energy-aware scheduling, as well as a comparison with the latest theoretical speed-scaling work.

2.1 Cloud Computing Infrastructure

A cloud computing system can be defined as an Internet-based development that aims to provide on-demand services. Its underlying framework is represented in different forms, one of which is the Infrastructure as a Service (IaaS) that we generally consider in this thesis. The innovative idea of IaaS was established in 2006 by Amazon Web Services [17], under the name of Elastic Compute Cloud (EC2), as an investment in their free or unused resources. In principle, IaaS is dedicated to dealing with hardware, software, servers, storage, and many other infrastructure components that are typically hosted by providers who take the responsibility of handling the affairs of regular maintenances and/or backup requirements. One of IaaS's advantages is that it provides a practical environment for treating requests that need highly scalable resources. This specific environment can be effortlessly customised on-demand, offering a tangible benefit to both end-users and service providers.

A multi-cloud system intuitively follows the distributed computing system that combines some features of centralised and decentralised computing systems. To clarify the architecture behind distributed multi-cloud systems, Table 2.1 gives a general comparison between centralised and decentralised computing systems, in terms of their pros and cons. Here, a distributed computing system, at an abstract level, can be defined as a system that consists of a set of individual computing nodes (i.e., clouds in the context of this thesis) that communicate via a network, in a collaborative way, to solve a single large problem. This system mostly follows a master/slave architecture, where unidirectional control over one or more nodes is done by a single master node [106]. Two of the key advantages

	Centralised	Decentralised
Pros	• A central computing node controls all devices and performs complex computations.	• Multiple computing nodes responsible for all processing.
	• A simple system to implement and manage.	• Has no single point of failure.
	• Has no concerns about any communication problems	• Enables distributing work and decisions which allows better performance.
	• Highly private and secure.	• Highly distributed and scalable.
Cons	• Has a single point of failure.	• A complex system to implement and manage.
	• Limited scalability.	• Synchronisation/communication among distributed nodes is critical.
		• All nodes have no view of the entire state at all times.
		• Security is an issue especially when using public network.

Table 2.1: Pros and cons of centralised and decentralised computing systems [78].

of the distributed computing system are its considerably good performance and power of computation. Big Data Hadoop, Google web services, Nebula, World Wide Web, etc., are examples of distributed computing systems.

The system architecture, followed in this thesis, is a distributed decentralised multi-cloud computing system. It represents a cooperative model of sharing resources that belong to different administrative clouds where these resources are mostly highly dispersed. This architecture, the so-called inter-cloud, is proposed with many different models, e.g. [104, 66, 31, 108], to serve as a federated approach to expand the resource pool and achieve great benefits with minor effort. In fact, the establishment of resource sharing between various Infrastructure Providers (InPs) would improve system performance, reduce the requirement to build more data-centres, enable the use of heterogeneity among multiple data-centres, and exploit the spare capacity in each data-centre especially during times of reduced demand. Thus, it is mooted that the future of the cloud computing system will likely be paying particular attention to the decentralised computing systems [84]. This has motivated us to consider primarily the distributed decentralised multi-cloud computing system as a promising model for our approach.

2.2 Energy Efficiency in the Cloud System

The efficient use of energy in the cloud system is aimed at reducing the amount of energy required to operate the cloud infrastructure including a large number of HPC servers and massive storage devices, etc., as well as in the connected inter-cloud for data transmission activities. Although cloud computing brought better solutions for Information Technologies (ITs) and individual users by providing convenient services and environments for applications to be executed efficiently in low-cost hosting in addition to all its other desirable features, it has turned energy wasting and overconsumption issues into major concerns. Further, there is an increasing need for cloud computing in multiple sectors, e.g., the evolution of eScience, Gaming systems and eCommerce. They are driven by high performance applications such as scientific computing, design and manufacturing, and big data applications such as in social data analysis and data mining applications. The changes in both sides of the equation – the demand of high-performance ubiquitous cloud computing, and the supply of distributed infrastructures – have a huge effect on energy demand and usage. Thus, two of the challenges faced by the next generation of high performance computing are power consumption and resource pool capacity. Concerning the power consumption problem, the desired goal of designing computing systems has been expanded to not only focus on improving the performance but to also support energy optimisation. These changes seem to threaten Moore's law by slowdown, after about 50 years of life [46].

Narrowing the discussion down to the main components responsible for the overall energy consumptions in a cloud environment, we briefly review the power consumption aspects of these components and their supported energy-saving techniques, as follows.

- The *server* is the major energy component in data-centres due to its processors and memories. For instance, the *processor* itself in a high-performance server consumes a significant amount of energy, accounting for up to 50% of the total energy consumption, which puts it in the first place, followed by the memory in the second place [55]. The annual energy consumption by a single high-performance 300-Watt server is 2628 KWh [23]. Regarding the energy efficiency, reducing the processor's energy consumption has the additional advantage of positively affecting the energy efficiency of other components such as the memories and buses [87]. Further, the recent energy-aware memory chip architectures utilise some spin down states to manage the energy consumed by the memory. The applied energy saving techniques for servers and processors include: deactivating the idle devices (also known as dynamic shutdown servers), Dynamic Voltage and Frequency Scaling (DVFS), virtualisation which supports migration and consolidation, and energy optimisation algorithms in general. We extend the discussion about the relevant energy-saving techniques for the computing server in more detail in Section 2.2.1.

- The *network* for the data communication includes the connectivity within the data-centre among various components e.g. servers and storage devices, and that between different data-centres of the same cloud. The main network components include switches, routers, and links. The network within the data-centre consumes about 30% of the total computational energy consumption [57]. The applied energy-saving techniques include: sleep mode, green routing, adaptive link rate, and virtual network embedding, and utilising technologies such as commodity electrical network elements, optical technology and hybrid network technology.

- The *cooling system* is an important component that is used to keep the whole data-centre elements and equipment at the right temperature by removing the heat produced by computational devices to avoid any damage and disruption. Thus, data-centre operators try to achieve this at all costs. Due to the need for a continuous, efficient and reliable cooling system, it is expected that this system will consume a relatively high amount of power to operate at such a level. Using renewable energy resources and building data-centres in cold areas are two of the approaches used to have a green cooling system.

2.2.1 Energy-Saving Techniques

Many studies have been conducted to investigate various energy management and saving techniques at hardware and software levels. We focus, in this section, on the existing software techniques for optimising the energy consumed by computing servers and processors, as follows.

Dynamic Voltage and Frequency Scaling (DVFS) is a technique that allows the processor to run at adjustable levels of frequency and voltage in a way that enables the dynamic management of the dissipated power, where the lower the voltage is the more frequency is sacrificed. As a dominant energy-saving technique, DVFS is applied in different models and environments [67]. Furthermore, the processor can support either *discrete* or *continuous* levels of frequencies [85]. When using the latter, the required frequency is chosen from a given range, e.g., $[f_{min}, f_{max}]$ where $f_{min} > 0$ is the minimum frequency and f_{max} is the maximum frequency; the chosen frequency, f, can take on infinitely many uncountable values from this range. However, in practice, the current processors only support a finite number of discrete frequency levels. This number is subject to increase, supported by technologies such as Intel SpeedStep and AMD PowerNow. To expand further, [47] give a general review of DVFS technique along with the consumption of the static and dynamic energy.

Broadly speaking, DVFS technique is highly effective in optimising, on the fly, both the frequency and the voltage, and it becomes increasingly significant in reducing energy loss and waste up to 34% at a single server CPU as discussed in [11] (cited in [83]). Although scaling down the CPU frequency could reduce the power consumption while meeting the targeted performance requirements of the application, some low frequencies can result in higher energy consumption if the computation takes a long time to complete. Therefore the power reduction using DVFS is not always optimal, and scaling well the frequency in accordance with specific requirements (e.g., demanding a high-performance computation or cutting energy costs/bills) is very complex and requires efficient algorithms to be managed.

Changing the state of idle servers is a Dynamic Energy Management (DEM) mechanism that can be equipped for considerable energy optimisation. Race to idle is a suggested technique, which combines speed scaling and power-state by turning the processor state to sleep after tasks have been executed at a high speed [1]. This technique has been studied and implemented by IBM [40]. However, there are some issues regarding the impact of the server off/on cycle on component reliability due to wear-and-tear [40].

The virtualisation technique with its high capabilities is, to a large extent, the core factor behind efficient resource utilisation. It is the process of creating virtual machines on the intended physical resources. This technique provides an effective way to save energy as it enables a reduction in the number of active physical machines (relying on Virtual Machine Migration and Consolidation techniques [73]). This technique, in turn, enables the sharing of bounded resources by virtually creating further machines or CPUs to potentially handle a huge number of computational workloads.

The algorithmic techniques for optimising energy in computing devices have been incorporated to manage and adjust the processor speed and/or the server state whenever appropriate to maximise their energy saving, for example, using a heuristic approach with the energy-aware scheduling of tasks/applications subject to deadline constraints. Many different approaches have been studied and applied to different platforms and environments, e.g., [20, 19, 67].

2.2.2 Data Computing and Transmission Energy

Consider an application app_m which is submitted to a service provider (an original cloud in the inter-cloud system). We focus, in this thesis, on two aspects of the energy consumed by app_m: the execution activities and the dataset transmission between clouds (if the original cloud is not the executer one). These aspects are discussed as follows.

Energy of execution activities: The energy consumption of a processor is affected by both execution activities $E_{dynamic}$, i.e., dynamic energy and leakage current, and static energy use E_{static}. As previously mentioned, the processor energy consumption can be controlled by employing the DVFS technique, where the lower the processor frequency is, the less instantaneous energy it consumes, but it incurs a longer execution time.

The conventional energy consumption metric of a single-core CPU running at frequency f is $E_{CPU} = \beta + \alpha f^3$ where β represents E_{static} and αf^3 represents $E_{dynamic}$ [41, 111, 53]. Thus, heterogeneous processors have different values of β and α. In this study, we adapt this formula to calculate the energy consumed by multi-core processors, discussed in Section 4.1.2.

Energy of dataset transmission: Data transmission is the transfer process of data as digital (electromagnetic) signals via a communication channel from one point to another over a network. The channel can be a wireless communication channel that carries signals like radio-wave or micro-wave; or a wired communication channel such as copper cables that carry electrical voltage signals, or fiber-optic cables that carry, e.g., light or infrared pulses. In multi-cloud computing system, data transmission is performed via the Internet.

In spite of the increasing use of data transmission by various cloud computing services, attention has not often been paid to the energy used in transmitting the data. Specifically, there are two types of data transmitting energy: one between various clouds in the inter-cloud network, and the other between each data-centre and the connected front-end computers (users). It is found that the latter type consumes even larger amounts of energy than storing the data with regular access [13]. In this study, we consider the optimisation of data transmitting energy between clouds combined with the computing energy consumption. While the computing energy and the data transmitting energy dominate the total energy consumed by a typical data-centre, it is impossible to confirm with certainty that the transmitting energy consumption and the computing energy consumption are always of the same order of magnitude or if one uses always more energy than the other. However, this mainly depends on the type of applications (tasks). For example, a video stream task would need an amount of data transmitting energy more than the required amount of computing energy and vice versa with a decryption task of a small data-size but a large computing-size.

2.3 Scheduling Concepts

The scheduling problem represents the process of making the decision of assigning a task (job), to be carried out, to one or more processors (or computational resources, in general). The problem involves a set of elements: a task that includes a set of operations with/without specified characteristics, resources such as machines, constraints such as time limits (deadlines), and an objective that represents the optimisation criteria with regard to some constraints and task characteristics [117].

2.3.1 Definitions

Before delving into our scheduling approaches, it is first necessary to understand various scheduling problems by distinguishing between some relevant concepts. These include task characteristics, e.g., preemptive or non-preemptive; problem constraints, e.g., online or offline; and scheduling styles, e.g., advance-reservation or best-effort (non-advance-reservation).

Preemptive scheduling represents the variant in which the execution of a job can be interrupted and resumed later on the same processor (no migration) or on an arbitrary processor (if migration is allowed). However, in non-preemptive scheduling once a job starts its execution, it must be run to completion without interruption. The difference between offline and online scheduling is distinguishable by whether the input instance (tasks) is known in advance (offline) or not, i.e. it is just presented when the scheduling is due (online).

Furthermore, there are two scheduling modes: best-effort scheduling and advance-reservation scheduling. In the best-effort scheduling mode, the scheduling decision is made depending on the instant status of the resources, and there is no guarantee of their availability. Applying this mode with some deadline-based task scheduling can be highly critical when a high number of parallel tasks need to be executed at the same time. However, the advance-reservation scheduler depends principally on the actual available slots for a given period of future time. It needs to ensure that the time frame of a task/application is scheduled within the resource capacity, considering both occupied and reserved resources.

2.3.2 Robust Scheduling Under Uncertainty

The scheduling under uncertainty problem, i.e., dealing with unknown parameters or unavailable information, is a very common and widespread concern in many models, especially when the objective is to optimise a scheduling process in distributed systems. Such optimisation is normally associated with uncertain timing parameters (e.g., resource availability with respect to time, processing times, makespan (expected performance), and deadline constraints, etc.,) and/or quality parameters like resource reliability or budget constraints [35, 98]. These parameters are always susceptible to unexpected deviations. Unlike the scheduling optimisation problem in a deterministic model, uncertainties in process scheduling require a systematic method and metrics to calculate in order to control the desire performance.

The scheduling approaches presented in this thesis have mainly three classes of uncertainties as follows:

- Uncertainty in processing times that ensure meeting the deadline. This applies to only our Best-effort approach, introduced in Chapter 4
- Uncertainty in successfully executing applications, where cloud resources are subject to unexpected failures.
- Uncertainty in optimising the overall energy consumption with tight application deadlines.

The literature on scheduling techniques under uncertainties conceptually concentrates on either: (1) enabling scheduling modifications on time whenever unexpected deviations occur, or (2) designing a robust approach that generates policies to be considered prior to the scheduling process. These two techniques are known as *reactive scheduling* and *preventive scheduling* respectively [62].

Approaches, in reactive scheduling, may include dynamic repair and rescheduling solutions to address uncertain parameters. Preventive scheduling that is followed by this thesis, however, includes several different methods to deal with the problem of uncertainty, such as stochastic based approaches, fuzzy programming, sensitivity analysis, and robust optimisation which is the relevant method to this thesis [86].

Robust scheduling, in the context of this thesis, aims to design a preventive scheduling to optimise energy consumption on the one hand, and to minimise the effects of unexpected resource failures that may violate application deadlines on the other hand, while maintaining the desired level of schedule performance. We propose specific robustness metrics to measure the utilised resources, while minimising energy consumption, in our multi-cloud system, attempting to balance utilisation over distributed heterogeneous clouds, introduced in Section 4.2. Our probability measurement relies on the standard deviation (SD), which is one of the most widely used metrics for assessing the robustness of a schedule [86]. In the remaining parts of this thesis, we will refer to our proposed criteria as Preference Rate Strategy (PRS) and Combination Rate Strategy (CRS).

2.3.3 Scheduling Complexity

In computational complexity theory, many computational problem are in the complexity class P (if it is possible to find their optimal solution in polynomial time), NP-complete, or NP-hard. The classification into the latter class needs to be proved by starting from scratch, which is very difficult, or by using an already proved NP-complete problem, which is called reduction. For some NP-hard problems, computing an optimal solution can work in reasonable time for very small instances based on exact optimisation techniques such as integer linear programming or branch-and-bound. However, for some real-world combinatorial optimisation problem instances that arise in practice, using a greedy approach or a heuristic approach is practically possible to solve them without ensuring an optimal solution.

In the context of scheduling, it is shown that many scheduling problems are NP-hard, and getting solutions close to optimal is of interest by developing polynomial time algorithms [97]. Further, when they are considered as decision problems, it is proved that many scheduling problems are NP-complete [27]. For the scheduling problem of this study, we use a heuristic approach and the competitive analysis technique to find a good solution and measure its quality. We give further discussion regarding the competitive analysis technique in the next section.

2.3.4 Online Speed-Scaling and Competitiveness

The theoretical study of online algorithms aims to measure the quality (expected cost) of the solution produced by an algorithm on any input sequence, by the use of competitive analysis approach [25]. This approach is regarded as pessimistic, since it considers the worst-case measurement. In other words, it is the approach in which the performance of the online algorithm is compared to the performance of its optimal offline algorithm OPT, where the online algorithm ALG is c-competitive if for all finite input sequence I, $ALG(I) \leq c \cdot OPT(I)$ [25]. In the speed-scaling problem, the DVFS is used to minimise the energy consumption by means of dynamically adjusting the speed at which a processor runs, where the rate of energy consumption is higher at higher speeds. The rate at which the energy is consumed is called the *power*. It can be represented by a function $f(s) = s^{\alpha}$, for some constant $\alpha > 1$, that maps the speed s to the rate of energy consumption. We refer to such functions as *standard power functions*.

The analysis of the scheduling problem that we consider at the level of HPC-applications subject to the deadline and precedence constraints is very difficult.

Thus, in Chapter 6 we set off from a non-preemptive version of the classical speed-scaling scheduling problem that was studied by Yao et al. [118]. This initial attempt of the analysis represents a step towards bridging the gap between the theoretical and the practical perspective of our problem and estimating how far from the optimal the used allocation method possibly is.

The scheduling problem of minimising the total energy consumption on a single processor using speed scaling was first posed by Weiser et al. [113] who studied different heuristics experimentally. The pioneering work by Yao et al. [118] analysed algorithms for speed scaling on a single processor with the objective of minimising the total energy consumption. Each job is identified by its release time, its deadline, and its work. It must be scheduled during the interval between its release time and its deadline, and preemption is allowed. The speed of the processor can be adjusted arbitrarily at any time. They presented a polynomial-time optimal algorithm for the offline problem and two online algorithms, Optimal Available (OA) and AVerage Rate (AVR). They showed that the competitive ratio of AVR is at most $\alpha^\alpha 2^{\alpha-1}$. In the theoretical part of this study, introduced in Chapter 6, we use a non-preemptive variation of AVR to schedule the jobs that are assigned to a processor.

2.4 Related Work and Comparison to the State of the Art

The task-scheduling problems taking into consideration energy-efficiency have been a hot subject of extensive technical and analytical research. Many technical approaches have been surveyed recently in [96, 73, 90] and other analytical approaches are reviewed in [14, 56].

Regarding our technical approach, the biggest difference between all the existing approaches and ours is that we consider:

- the energy cost of transferring datasets when globally scheduling applications/tasks over geographically distributed clouds; and
- the occupation rate of cloud resources as a factor to minimise application deadline violations.

The novelty here as far as this thesis is concerned is to precisely schedule whole applications or individual dependent tasks, based on a combination of resource occupation and two energy dimensions: energy consumed for tasks execution and data transmission. Apart from considering transmission energy, many approaches have been suggested to address the objective of minimising energy consumption from different perspectives such as data-centre management architecture [60, 94], scheduling workflows [68], reservation in mobile networks [109], or scheduling tasks in mixed-criticality systems [65].

In this section, we limit our discussion to the closely relevant approaches in terms of applying their scheduling in decentralised systems, e.g., [102, 9, 66, 107, 84], or considering the energy consumption problem for scheduling HPC applications/tasks in cloud computing systems, e.g., [68, 116, 71]. Table 2.2 summarises in detail these approaches. Following that, this section will focus on the theoretical approach of the speed-scaling problem. Here, we present the current state of the art in homogeneous and heterogeneous parallel processors settings to highlight our theoretical contribution.

				Feature Support								
	Environment	**Optimisation Objective/s**	**Optimisation Method/s**	Online	Offline	DVFS	Consolidation	Rescheduling	Virtualisation	Migration	DAG-apps	**Evaluation Method**
A	- Decentralised - Heterogeneous multi-clouds system.	Minimising execution energy, dataset transmission energy consumption subject to the deadline constraint.	Set of heuristic scheduling algorithms with best-effort (EGSBE), and advance-reservation (EGSAR).	*		*			*		*	Our own simulation of multi-clouds system.
B	- Decentralised - Heterogeneous multi-processors network.	Minimising communication and load transfer overheads.	Load scheduling/ balancing algorithm called ELISA.	*				*		*		Their own simulation of a multi-processor system.
C	- Decentralised - Heterogeneous Grid system.	Minimising communication and load transfer overheads.	Load balancing algorithms: - LBA for small size Grids. - Modified ELISA for large-scale Grids.	*				*		*		Their own simulation of a multi-processor system.
D	- Decentralised - Heterogeneous Grids and clouds system.	Optimised scheduling performance (e.g. job slowdown, job waiting time) and overheads (e.g. generated/ transferred messages)	Community aware scheduling algorithm (CASA).	*				*		*		The MaGateSim simulator.
E	- Decentralised - Heterogeneous Grids and clouds system.	Optimised scheduling performance and execution energy consumption.	Cooperative Scheduling Anti-load balancing Algorithm for Cloud (CSAAC).	*			*	*		*		Extended version of MaGateSim simulator.
F	- Decentralised - Heterogeneous multi-clouds system.	Optimised scheduling performance (e.g. task execution time).	Cloud Min-Min Scheduling (CMMS) algorithm.	*					*		*	Their own simulation environment of IaaS clouds system.
G	- Centralised - Heterogeneous clouds system.	Optimising makespan, execution and cooling energy consumption, and deadline violations	MultiObjective Evolutionary Algorithms (MOEAs).		*						*	Their own simulation environment of a data-centre.
H	A heterogeneous cloud data-centre.	Optimising execution energy consumption	A heuristic scheduling algorithm	*		*			*			CloudSim simulation.
I	A heterogeneous cloud data-centre.	Optimising a bi-objective function of either energy consumption or makespan.	A Multi-Heuristic Resource Allocation (MHRA) algorithm.	*					*		*	A private cloud hosted by the Barcelona Super-computing Centre (BSC) consists of two types of computing nodes.

				Feature Support								
	Environment	**Optimisation Objective/s**	**Optimisation Method/s**	Online	Offline	DVFS	Consolidation	Rescheduling	Virtualisation	Migration	DAG-apps	**Evaluation Method**
J	- Centralised - Heterogeneous data-centres system.	Minimising carbon emissions or maximising profit subject to the deadline constraint.	Set of heuristic scheduling policies.	*		*		*			*	Their own simulation of 8 data-centres.
K	- Centralised - Data-centres system.	Minimise energy cost and carbon emissions subject to the deadline constraint.	A heuristic workflow scheduling Algorithm.	*		*			*		*	CloudSim Simulation.

References of the mentioned works

A	Our approach, presented in Chapters (3-6)
B	[9] - Anand et al. (1999)
C	[102] - Shah et al. (2007)
D	[66] - Huang et al. (2013)
E	[107] - Thiam et al. (2013)
F	[84] - Li et al. (2012)
G	[68] - Iturriaga et al. (2016)
H	[116] - Wu et al. (2014)
I	[71] - Juarez et al. (2016)
J	[53] - Garg et al. (2009)
K	[36] - Cao et al. (2013)

Table 2.2: A summarised comparison between the relevant approaches to this thesis

2.4.1 Scheduling in Decentralised Systems

There has been a lot of research on addressing scheduling problems in decentralised computing systems [9, 102, 66, 107, 84, 105]. Their optimisation methods, in general, are applied at different levels ranging from high-performance processor systems up to decentralised cloud or grid systems.

At the processor system level, Shah et al. [102] suggest a modification of the Estimated Load Information Scheduling Algorithm (ELISA) [9] for reducing the communication delay and load transfer cost among multiple processors, the so-called Modified ELISA scheduling algorithm (MELISA), in addition to a Load

Balancing on Arrival scheduling algorithm (LBA). Unlike LBA, MELISA seems to scale better to large systems. The similarity among these algorithms (ELISA, MELISA and LBA) is that they are all applied at the level of the processor network systems. However, ELISA is not efficiently applicable in heterogeneous grid-based systems, in contrast to MELISA [102].

At high-level systems, using rescheduling, Huang et al. [66] introduce a Community Aware Scheduling Algorithm (CASA) as a decentralised dynamic scheduling approach. It is comprised of five heuristic sub-algorithms that work in two phases: the job submission phase and the dynamic scheduling phase. In addition, CASA has been extended by a Cooperative Scheduling Anti-load balancing Algorithm for Cloud (CSAAC) in [107]. CSAAC applies a consolidation mechanism for energy-efficient scheduling. Thereby, the slowest task is migrated from an overloaded host to the host with minimum energy consumed and best execution time. From our perspective, adopting the iterative rescheduling method to address our problem may affect the scheduling efficiency due to the large cost and the even larger overhead of transferring large datasets (for task dependencies).

In an anti-rescheduling manner, the Cloud Min-Min Scheduling (CMMS) algorithm, proposed by Li et al. [84], is applied to multi-cloud systems and inspired by the popular min-min algorithm. The CMMS scheduler maintains the dependency constraints with every scheduling step, since it partitions HPC-applications to schedule each task individually. It relies on pulling information about resource status from participant clouds in order to select a cloud that offers the minimum estimated earliest finish time for each mappable (ready) task. Then, it allocates all mappable tasks according to their minimum estimated finish time. The CMMS scheduling approach [84] is similar to ours in terms of scheduling HPC-applications in decentralised multi-cloud system, but different in terms of the main objective towards performance optimisation. We implement the CMMS for comparison purposes with our proposed global schedulers, i.e. EGSBE and EGSAR (discussed in Chapter 4 and Chapter 5 respectively).

Sotiriadis et al. [105] propose an algorithmic structure of four collaborating algorithms (ICMS), applied in the inter-clouds system, to serve requests for web services. Their approach models a set of interconnected clouds each of which can have more than one data-centre that consists of a set of machines called hosts. In their approach, a service request with specified requirements that cannot be fulfilled locally is distributed to interconnected clouds where the first responder that is able to carry out the request is chosen. Here, each request for job delegation is iteratively distributed until a successful delegation is achieved. Further, the number of iterations is set by default to 3, where the job is terminated in the case of continuous Service Level Agreement (SLA) mismatching. When the job is assigned to the chosen host based on some performance criteria beside the SLA matchmaking, a VM is created and then allocated to the host (machine). The performance can be affected by the percentage of the machine that is dedicated to the VM. Regardless of the energy inefficiency of this approach, it has many limitations regarding the lack of its applicability to schedule HPC tasks because of the extensive sharing of physical machine (high use of virtualisation) that accordingly reduce the execution performance.

Sotiriadis et al. [104] evaluate some pre-2012 works that apply a meta-scheduler in highly dynamic settings. Their evaluation aims to check the applicability of the existing approaches to the inter-cloud framework in terms of a set of requirements that are suggested as needs for inter-cloud systems. Specifically, a candidate work should provide a good support for unpredictability management, heterogeneity, geographical distribution, variation of job requirements, SLA compatibility and rescheduling. From this perspective, our approach can be seen as a high candidate that supports some requirements. More precisely, the global scheduler supports resource heterogeneity, geographical distribution and variation of job requirements, and the SLAs can be enabled at some point.

To complete the picture of the decentralised system, an efficient communication or remote resource monitoring method is needed to keep a reasonable level of consistency between distributed entities without directly accessing the remote resources. To this end, there are several mechanisms for remote resource monitoring that can be integrated with the scheduling policies (meta-schedulers) in inter-cloud systems. For example, Huang and Wang [64] propose a push and pull (P&P) model that extends the common monitoring method from grid and cluster computing that is called the push model and the pull model. In the grid and cluster computing, there is a producer that produces the resource status and a consumer that consumes the information from the producer. In the pull model, the consumer pulls the resources status from the producer and if any update occurs in the producer side it triggers the push model to push the new resources status to the consumer. The P&P model exploits the advantages from both push model and pull model and enables adjustment to be made between the two models as well as the number of updates to be modified according to the user's needs. This model is adopted in [84], discussed above, as well as in this thesis.

The message passing mechanism, proposed by Bessis et al. [22], is another communication method to cope with the drawbacks of the traditional solutions, e.g., flooding of the message-exchange system. In this mechanism, a requester sends query messages to all connected entities and receives messages only from available entities. This is to optimise the energy efficiency for message exchanging in the inter-cloud network. However, from our perspective, both request messages and response messages are mainly very small in size, and so their transmission cost may be negligible. This mechanism is adopted by Sotiriadis et al. [105], discussed above.

2.4.2 Green Scheduling in Multi-Cloud System

Energy-aware scheduling, generally, aims to solve a scheduling problem while considering energy-consumption minimisation, including different objectives and constraints, e.g., in approaches like [76, 74], and more are surveyed in [19]. The following concentrates on the most relevant work applying the DVFS energy-saving technique.

Like in our technical approach, Iturriaga et al. [68] consider the scheduling of HPC workflows (dependent tasks) with deadline constraints and also the conceptual use of scheduling levels: *global* and *local*. The *global* scheduling is applied to a distributed cloud system – their cloud system is centralised while ours is decentralised – to assign workflows to data-centres, whereas the *local* scheduling is for mapping tasks to machines. They use offline MultiObjective Evolutionary Algorithms (MOEAs) with the objective of optimising makespan, energy consumption, and deadline violations. Here, deadline violation can be accidentally caused by resource failures (e.g., when servers or network communications are down for maintenance) or by imprecise scheduling decisions due to the distributed environment. The latter is not expected to occur by our AR-scheduler, since we adopt a token-based reservation schedule, discussed in Chapter 5. However, in the case of our BE-scheduler, see Chapter 4, we tried, to some extent, to cope with such problem by utilising our proposed Combination Rate Strategy (CRS) to minimise deadline violations by considering the resource occupation rate. It should be mentioned that the dynamic environment with non-guaranteed resource availability is prone to deadline violation, especially with respect to the dependency constraints where data transmission over an inter-cloud network can be time consuming.

Wu et al. [116] propose a heuristic scheduling algorithm for heterogeneous computing environments, aiming to minimise power consumption without influencing the performance in order to satisfy the SLA. In their approach, the minimum and maximum frequencies need to be specified with submitted tasks as well as the SLA. For a given task, the scheduler repeatedly creates and assigns VMs from servers that remain within the required performance range until the task requirement is satisfied. If the requirement is still not satisfied, the scheduler turns on idle servers (if they exist) as required to continue the creation and assignment process of VMs. Their method clearly has associated overhead costs when the scheduling fails. On top of this, creating or destroying VMs consumes non-trivial energy, as Juarez et al. [71] demonstrated. Additionally, providing thresholds for the required frequency, particularly the maximum one, may limit the performance, which seems critical for deadline-based applications. Compared to their method, our approach differs in that our local scheduler (ELS), introduced in Chapter 4, determines the best possible frequency to execute each task from the whole range of frequencies that is supported by the processors. We rely on both the provided computing volume per VM and the required number of machines for each task, while ensuring not to violate the deadline constraint of the whole application in the case of AR mode, and each task in the case of BE mode.

A scheduling approach for optimising a bi-objective function of either energy consumption or makespan was proposed by Juarez et al. [71] for heterogeneous cloud systems. They provide a combined cost function with a weighting factor α that indicates the user preference of either choosing energy-efficiency or execution time. Their heuristic algorithm ranks tasks of a given Directed Acyclic Graph (DAG) by estimating the required energy. This is to determine independent subsets of tasks as a preparation step before allocating resources. In their method, the consumed energy is estimated by multiplying the task processing time by the proportional mean power. Compared to this, our energy model utilises DVFS where task execution time is multiplied by its instantaneous consumed energy, which comprises both the static and the dynamic energy. The decision of our global scheduler relies on one of the proposed strategies:

preference rate or *combination rate* strategy, see Section 4.2. The latter aims to minimise applications/tasks deadline violations caused by resource failures alongside energy optimisation.

Garg et al. [53] suggest scheduling policies that can be applied in heterogeneous data-centres that belong to a single cloud provider. These policies represent centralised meta-schedulers and rely on periodically updated information from both cloud sites and users whenever assigning a workflow to a cloud site. Policies such as minimising carbon emissions or maximising profit are used according to the objectives of the cloud provider. They also design a DVFS-based local scheduler whereby the energy consumption can be minimised within their data-centres by scaling up/down the frequency of the processors from a discrete range of speeds. For a submitted application, the scheduler chooses the frequency based on a pre-calculated value at the meta-scheduler level, called optimal frequency $f_{opt} = \sqrt[3]{\frac{\beta}{2\alpha}}$. In the case that f_{opt} does not belong to the discrete range of supported frequencies, the allocated machines run at the nearest frequency to f_{opt}. However, this speed may not be the optimal one to meet the application deadline and, in such a case, the scheduler needs to increase the speed and try to find a free slot to execute the application, in a rescheduling manner. In our approach, the local scheduler calculates the best speed, in terms of minimising energy while meeting the deadline, to execute each task in the application without any future update to such speed. We expect that our greedy method can produce better results, in terms of avoiding deadline violation and reducing energy consumption, than executing at a lower speed first before increasing the speed later.

In a similar vein, Cao et al. [36] propose a heuristic workflow scheduling algorithm for a centralised cloud system that functions in three steps. The first step decides the workflow acceptance. The second step uses the DVFS technique to minimise energy subject to the deadline constraints, by choosing an optimal frequency for each task. They adopted the calculation of the optimal frequency from [53]. The third step performs a backward task allocation to the VMs in order to minimise the VM overhead by reusing them whenever it is possible as well as reducing machine idle time when appropriate. This is like our approach in utilising the DVFS technique in the first two steps. However, their mechanism for allocating VMs would not be beneficial for our best-effort scheduling scenario, since backward allocation cannot be applied due to the non-reservation manner where the scheduling decision relies on the instantaneous state of the resources. On the other hand, it can be integrated with our AR approach as a local scheduling policy.

2.4.3 Performance Evaluation in Cloud Computing

In the previous section, we have reviewed concisely some relevant scheduling approaches in the area of the multi-cloud environment. As a wider discussion, this section will give a brief synopsis regarding the performance evaluation of scheduling problems (i.e., non-functional requirements) in general. To our knowledge, after having a wider look and review at the literature on the ideal methods used to evaluate the performance of a schedule, we imagine that there seems no universal consensus among researchers on an ideal metric or analytical technique for performance evaluation. Experiments to verify the desired performance can be carried out in a number of different techniques, including (1) *analytical modeling*, (2) *measurement data*, or (3) *software simulation based approaches* [70], discussed in some details as follows:

- Analytical modeling technique seems efficient in providing prompt answers as well as in helping to understand the interactions between involved parameters. However, this technique is often constrained by assumptions and simplifications that may make the outcomes too far from reality or practice. A queuing theory model is a well-known example of this technique, see [69], and also an example of applying this technique on cloud computing [110].

- Measurement data technique is usually applied by collecting data from a real environment, typically by on-site visit and/or gaining a remote access to the organisation's resources (e.g., a cloud data-centre). It effectively helps to discover unpredictable interactions between involved parameters, and always provide trusted and highly accurate results. In contrast to these positive aspects, this technique faces several challenges, notably, (1) it can be affected by the environment changes, and (2) it is not flexible for controlling parameters/variables and has to adapt to the existing configuration only.

- Software simulation technique, as compared to the measurement technique, is considered less expensive in terms of required equipment or the time needed to conduct the experiment. The drawbacks of this technique are that it is sometimes hard to test/debug the implementation of the simulation in the case of encountering accidental failures or in the case of obtaining inaccurate results. Another drawback (that is in line with the analytical modelling technique) is that simulation always works under many assumptions and simplifications, especially when defining so-called object-entities. We will give a further discussion on software simulation tools used for cloud computing in Section 3.3.

However, no best technique can be identified as the choice among those three methods depends on the considered framework of the approaches. Here, a framework describes aspects like field of the study, complexity of the approach, accessibility, and capability of observing system performance, etc. To clarify more, practical approaches that involve critical models/applications normally need to be measured, with respect to their objectives, using complex metrics or simulations, whilst objectives of other normal approaches (e.g., non-critical/non-commercial approaches) can be well-evaluated using some specific metrics. These specific metrics are often constrained by assumptions, and they allow one to assess the effects of parameters that are associated with the performance (e.g., time) from specific or limited angles. The evaluation method followed by this thesis is based on software simulation, discussed in Chapter 3, and that is due to two factors:

- (1) the measurement and analysis techniques on multi-cloud environments are physically impossible or too expensive to conduct; and
- (2) for best assessment of performance evaluation of cloud computing on real large-scale distributed platforms, software simulation comes first [52], as it enables to virtually experiment with proposals with an easy way of controlling variables and parameters.

Concerning the available metrics for performance evaluation, many researchers have studied different criteria, such as: job slowdown, job waiting time, and generated/transferred messages [66, 68]; task execution time [84]; energy usage [116, 71, 53, 36, 68]; makespan [68, 71]; deadline violation [68]. Additionally, many other quantity-metrics also exist that focus on the cost, throughput, time of allocation and release of resources, delay in service and productivity, number of I/O operations in the network, etc. The considered metrics in this thesis have been developed on top of our proposed energy model (cf. Section 4.1.2) and the level of resource utilisation (cf. Section 4.2).

2.4.4 Speed-Scaling on Parallel Processors

The classical problem of minimising the energy consumption on a single speed-scaling processor considered by Yao et al. [118] (cf. Section 2.3.4) has been extended to various other objectives such as minimization of energy plus flow time [81, 61]. An overview of known results up to 2016 can be found in the recent surveys by Bampis [14] and Gerards et al. [56]. The known results that are most relevant to this thesis are those that consider speed scaling for energy minimisation of jobs with release times and deadlines on both homogeneous and heterogeneous parallel processors. Such problems have mostly been studied in the preemptive case, and much less is known about the non-preemptive case.

On *homogeneous parallel processors*, a number of studies have been conducted on the preemptive case [2, 18, 4, 59] and some on the non-preemptive case [15, 43]. Albers et al. [4] show that the minimum energy scheduling problem with arbitrary release times and deadlines on m parallel processors allowing preemption is NP-hard even for unit size jobs but polynomial for unit size jobs with agreeable deadlines. They propose an offline algorithm for arbitrary size jobs with agreeable deadlines called Classified Round Robin (CRR). Their algorithm produces an approximation ratio of $\alpha^{\alpha}2^{4\alpha}$. They also adapt CRR to the online scenario and achieve a competitive ratio of $\alpha^{\alpha}2^{4\alpha}$ for unit-size jobs with arbitrary release times and deadlines, and for arbitrary size jobs with agreeable deadlines.

Later, Bell and Wong [18] show that, for arbitrary size jobs with arbitrary deadlines, CRR is no longer constant competitive. They present a variation called Dual-Classified Round Robin (DCRR) that has a competitive ratio of $2^{4\alpha}((\log P)^{\alpha} + \alpha^{\alpha}2^{\alpha-1})$ when applying AVR on each processor, where P is the ratio between the maximum and minimum job size. Greiner et al. [59] present a randomized B_{α}-approximation algorithm and a randomized $2(\frac{\alpha}{\alpha-1})e^{\alpha}B_{\alpha}$-competitive online algorithm, where B_{α} is the α-th Bell number. They use generic reductions from multi-processor to single-processor algorithms.

The problem with preemption and migration has been considered by Albers et al. [2] for jobs with arbitrary size and arbitrary release times and deadlines. They give an optimal offline algorithm and propose two online algorithms OA(m) and AVR(m), adapted from OA and AVR. They show that OA(m) and AVR(m) produce competitive ratios of α^α and $(2\alpha)^\alpha/2+1$, respectively. For homogeneous non-preemptive speed scaling, Cohen-Addad et al. [43] propose a $((\frac{w_{max}}{w_{min}})^\alpha(\frac{5}{2})^{\alpha-1}\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha)$-approximation algorithm for the offline problem of minimum energy scheduling, where $\tilde{B}_\alpha$ is the generalised Bell number. Bampis et al. [15] tackle the offline problem for both agreeable and general instances. They obtain ratio $(2-\frac{1}{m})^{\alpha-1}$ for agreeable instances and a ratio dependent on the number of processors for arbitrary instances. Their proposed algorithms follow the idea of converting a given preemptive solution into a feasible non-preemptive one. They also show that the ratio between the optimal non-preemptive schedule and the optimal preemptive one can be $\Omega(n^{\alpha-1})$.

On *heterogeneous parallel processors*, the speed-scaling problem has recently been studied in both preemptive [3] and non-preemptive [16] cases. Albers et al. [3] study the online version of the problem with migration and propose a $((1+\epsilon)(\alpha^\alpha 2^{\alpha-1}+1))$-competitive algorithm called H-AVR. It aims to assign work in each time interval (slot) according to the AVR schedule, and for each interval it creates an offline $(1+\epsilon)$-approximate schedule based on maximum flow computations.

Bampis et al. [16] tackle the offline non-preemptive version of the fully heterogeneous speed scaling problem, where the work of a job can be processor-dependent. They improve the approximation ratio of $(\frac{w_{max}}{w_{min}})^\alpha(\frac{5}{2})^{\alpha-1}\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$ for the homogeneous case from [43] by a factor of $(\frac{5}{2})^{\alpha-1}(\frac{w_{max}}{w_{min}})^\alpha$ while handling the fully heterogeneous case. Their proposed $\tilde{B}_\alpha((1+\epsilon)(1+\frac{w_{max}}{w_{min}}))^\alpha$-approximation algorithm combines (i) a $\tilde{B}_\alpha(1+\epsilon)^\alpha$-approximation algorithm for the preemptive, non-migration speed scaling problem on fully heterogeneous processors; and (ii) a $(1+\frac{w_{max}}{w_{min}})^\alpha$-approximation algorithm for the non-preemptive speed-scaling problem applied on a single processor. Although

Type	Problem	Ratio
Online	• $S \mid r_j, d_j, w_j = 1, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [4]
	• $S \mid agreeable, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [4]
	• $S \mid r_j, d_j, pmtn, no\text{-}mig \mid E$	$2^{4\alpha}((\log P)^\alpha + \alpha^\alpha 2^{\alpha-1})$ [18]
	• $S \mid r_j, d_j, pmtn, no\text{-}mig \mid E$	$2(\frac{\alpha}{\alpha-1})^\alpha e^\alpha B_\alpha$ (randomized) [59]
	• $S \mid r_j, d_j, pmtn, mig \mid E$	α^α [2]
	• $S^* \mid r_j, d_j, pmtn, mig \mid E$	$(1+\epsilon)(\alpha^\alpha 2^{\alpha-1} + 1)$ [3]
	• $S^* \mid r_j, d_j - r_j = x, \delta_j = \delta \mid E$	$\boldsymbol{3^{\alpha_m+1}(\alpha_m^{\alpha_m} 2^{\alpha_m-1} + 1)}$
	• $S^* \mid agreeable \mid E$	$\boldsymbol{5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m-1} + 1) \lceil \log D \rceil^{\alpha_m+1} \lceil \log T \rceil^{\alpha_m+1}}$
Offline	• $S \mid agreeable, pmtn, no\text{-}mig \mid E$	$\alpha^\alpha 2^{4\alpha}$ [4]
	• $S \mid agreeable \mid E$	$(2 - \frac{1}{m})^{\alpha-1}$ [15]
	• $S \mid r_j, d_j \mid E$	$(m^\alpha(\sqrt[m]{n}))^{\alpha-1}$ [15]
	• $S \mid r_j, d_j \mid E$	B_α (randomized) [59]
	• $S^* \mid r_j, d_j \mid E$	$\tilde{B}_\alpha((1+\epsilon)(1 + \frac{w_{max}}{w_{min}}))^\alpha$ [16]

TABLE 2.3: Known and new (in bold) results for speed-scaling on parallel processors. S stands for homogeneous and S^* for heterogeneous processors

the general case of the non-preemptive scheduling problem is NP-hard even on a single processor, the special case with agreeable deadlines can be solved in polynomial time for a single processor as there is an optimal preemptive schedule that does not use preemption, as observed e.g. in [10].

Table 2.3 shows known results for minimum energy speed scaling problems for jobs with release times and deadlines on both homogeneous (S) and heterogeneous (S^*) parallel processors, including our results (in bold, discussed in details in Chapter 6). The problems are identified using an adaptation of the standard three-field notation of Graham et al. [58]. The previously known results do not cover the online problem of non-preemptive speed-scaling on heterogeneous processors, which is considered in this thesis.

2.5 Summary

In this chapter, we have presented the concepts and definitions that are most relevant to our study, which merges the fields of decentralised multi-cloud systems and scheduling problems for the main goal of energy optimisation while respecting deadline constraints and resource reliability.

Specifically, this chapter has provided an overview of IaaS and reviewed the energy-efficiency issue of operating the cloud infrastructure. Definitions of some important concepts for scheduling problems have been provided, and a distinction has been drawn between preemptive and non-preemptive scheduling, online and offline scheduling, and best-effort and advance-reservation scheduling styles.

The chapter also included a discussion on the most related work in terms of scheduling in a decentralised environment and energy-aware scheduling, as well as a comparison of our results to the latest speed-scaling results from a theoretical perspective. In the next chapter, we will present in detail our system model that is implemented in a prototype tool for analysis and evaluation.

Chapter 3

A Simulation-Based Framework for Analysis and Evaluation

Cloud infrastructures are fundamentally data-centres, which often demand high energy usage to permanently operate [23]. Thus far, many common solutions suggested to efficiently schedule and/or run HPC-workloads have focused on minimising energy usage at a single cloud, but not across multiple clouds. To estimate the potential energy reduction on a global scale as in multi-cloud computing (i.e., geographically distributed in different locations around the world), we have built up a software simulation that mimics running HPC-workloads on a multi-cloud environment.

Before delving into our simulation, we devote the first part of this chapter to define the considered system model, including the abstract structure of our multi-cloud and HPC-application models. In the second part, we focus on the design and implementation of the proposed tool that depends principally on the discrete-event simulation technique. Here, we give some technical discussion on how our simulation can be used, and what the main functionalities/features are that one can exploit to analyse the scheduling results. We discuss, in the last part of this chapter, the relevant existing software simulations, explaining how our tool differs from them.

3.1 Proposed Models

The proposed system model suggests mainly a heterogeneous computing scheduling mechanism that becomes sufficiently important with the growth of many cloud services around the world. From an environmental improvement perspective, this model offers sharing multi-cloud resources that are controlled by different vendors for generally reducing computational energy consumption. Additionally, our model allows local energy aware scheduling of homogeneous cloud resources to reduce possible energy consumption. Concerning cloud applications, we model a generic HPC-workflow structure for parallel and distributed applications. This application model is typically composed of a set of dependent tasks (jobs), such that tasks may require the outputs from other tasks, allowing to abstractly design applications ranging from normal e-commerce web applications to high scale scientific applications [52]. Here, each task requires computation as well as data transmission activities. Furthermore, our application model intuitively implies the support of applications that consist of independent tasks. The main characteristics of the proposed multi-cloud model as well as the considered application model are discussed in more detail in Sections 3.1.1 and 3.1.2.

3.1.1 Multi-Cloud Model

In this research context, we consider a decentralised multi-cloud system that consists of a number of geographically distributed heterogeneous clouds, owned by different providers as illustrated in Figure 3.1. These clouds participate in a federated approach (cf. Section 1.1.2). The system, from the top level, consists of a set C of decentralised clouds, where $C = \{c_1, \cdots, c_k\}$, $k \in \mathbb{N}$. Each cloud c_j has a homogeneous data-centre, characterised by six parameters as described in Table 3.1. The manager server ms_j of each cloud relies on three components:

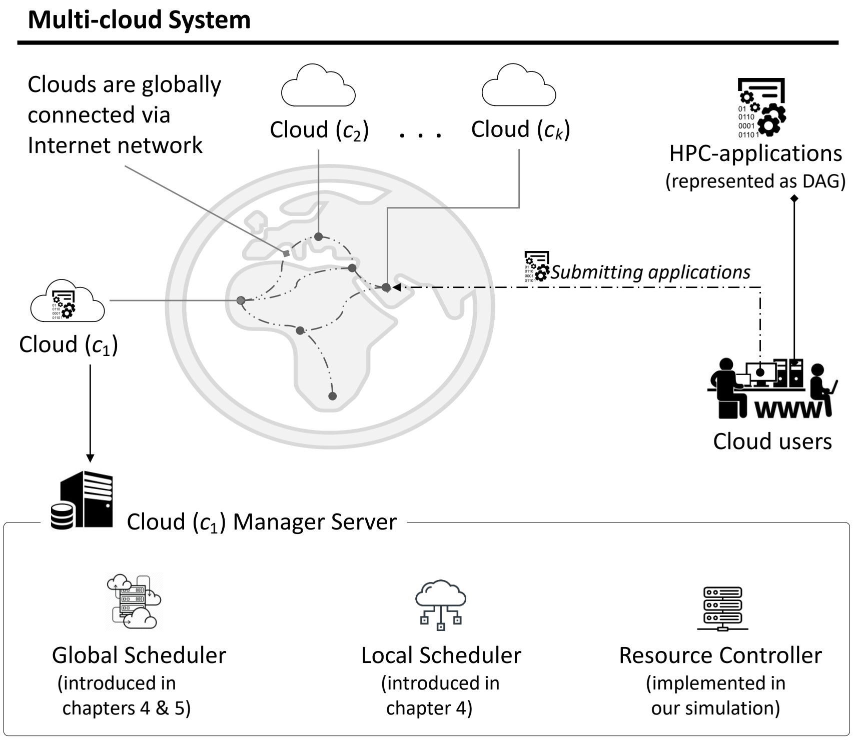

Multi-cloud System
Clouds are globally connected via Internet network
Cloud (c_2)
. . .
Cloud (c_k)
HPC-applications (represented as DAG)
Submitting applications
Cloud (c_1)
Cloud users
Cloud (c_1) Manager Server
Global Scheduler (introduced in chapters 4 & 5)
Local Scheduler (introduced in chapter 4)
Resource Controller (implemented in our simulation)

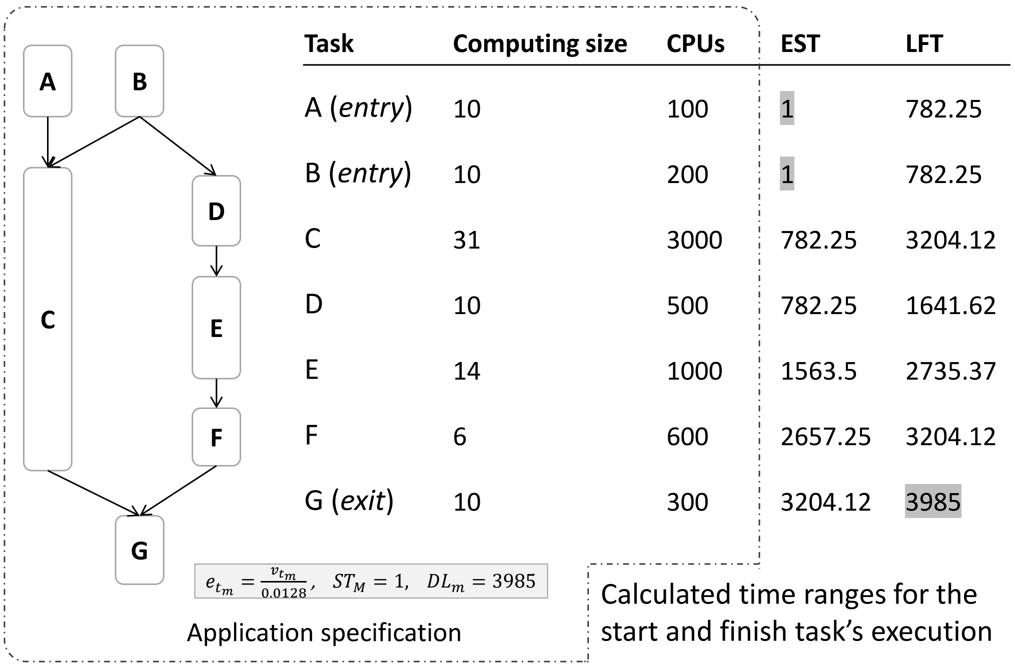
FIGURE 3.1: Proposed model representing distributed heterogeneous multi-cloud structure, which includes two scheduler components, discussed in detail in the mentioned chapters

.

Parameter	Description
ms_j	manager server
nS_j	number of servers
$nCPU_j$	number of physical processors per server
$capacity(s_{c_j})$	number of virtual machines per server
$[f_{min_j}, f_{max_j}]$	minimum and maximum frequencies of a processor
β_j and α_j	processor power parameters

TABLE 3.1: Cloud parameters of a homogeneous data-centre.

a *global scheduler*, a *local scheduler*, and a *resource controller*. The latter acts as a resource checker, and is also responsible for query messages with participating clouds.



Task	Computing size	CPUs	EST	LFT
A (*entry*)	10	100	1	782.25
B (*entry*)	10	200	1	782.25
C	31	3000	782.25	3204.12
D	10	500	782.25	1641.62
E	14	1000	1563.5	2735.37
F	6	600	2657.25	3204.12
G (*exit*)	10	300	3204.12	3985

$e_{t_m} = \frac{v_{t_m}}{0.0128}$, $ST_M = 1$, $DL_m = 3985$

A
B
C
D
E
F
G
Application specification
Calculated time ranges for the start and finish task's execution

FIGURE 3.2: An example of a simple DAG application, consisting of 7 tasks

3.1.2 HPC-Application Model

Consider the left part of Figure 3.2, it illustrates the structure of applications followed in this thesis as a directed acyclic graph (DAG). Here, an application $app_m = (V_m, E_m, ST_m, DL_m)$ consists of a set V_m of dependent tasks such that $V_m = \{t_1, \cdots, t_q\}$, $q \in \mathbb{N}$ and a set E_m of directed edges, each representing a data dependency between two tasks. The sets of direct predecessors and successors of a task t_i are denoted by $pred(t_i)$ and $succ(t_i)$, respectively. ST_m is the start time of app_m, and DL_m is the deadline. The original cloud of app_m is denoted by o_{app_m}, indicating the receiving cloud of the first submission of app_m.

There are two distinguished tasks in V_m, namely, *entry* and *exit* tasks, where both can be multiple in the application. As shown in Figure 3.2, a task t_i is considered *entry* (see the task A or B), if its $pred(t_i)$ is $\emptyset$, meaning that it has no predecessors, and *exit* (like the task G in Figure 3.2) if its $succ(t_i)$ is $\emptyset$. Each task $t_i(n_{t_i}, v_{t_i}, EST_{t_i}, LFT_{t_i}) \in V_m$ is defined by four parameters as follows: n_{t_i} is the required number of VMs, v_{t_i} is the computing volume per VM, EST_{t_i} is the task's earliest start time, and LFT_{t_i} is the task's latest finish time. As a guide for the global scheduler, EST_{t_i} and LFT_{t_i} of each task are calculated using (3.1) and (3.2), relying on start time ST_m and deadline DL_m of the submitted application.

$$EST_{t_i} = \begin{cases} ST_m & \text{if } t_i \text{ is entry} \\ \max_{t_m \in pred(t_i)}(EST_{t_m} + e_{t_m}) & \text{otherwise} \end{cases} \tag{3.1}$$

$$LFT_{t_i} = \begin{cases} DL_m & \text{if } t_i \text{ is exit} \\ \min_{t_m \in succ(t_i)}(LFT_{t_m} - e_{t_m}) & \text{otherwise} \end{cases} \tag{3.2}$$

Here, $e_{t_m} = \frac{v_{t_m}}{f}$ is the execution time of t_m at the minimum speed in a list of speeds that contains only the maximum speed of each cloud.

Figure 3.2 exemplifies a simple DAG application, consisting of 7 tasks, on which the variables EST_{t_i} and LFT_{t_i} can be calculated. Given such application specifications with the assumption that the maximum speed f to compute e_{t_m} is 0.0128, $ST_m = 1$, and $DL_m = 3985$, see the left part of Figure 3.2, we calculate the time ranges to start and finish executing each task using (3.1) and (3.2), such that all the tasks must be scheduled within their defined ranges.

3.2 MCST: Prototype of a Multi-Cloud Simulation Tool

Software simulation is generally used to analyse techniques that are physically impossible or too expensive to conduct. Rather than evaluating cloud computing on real large-scale distributed platforms that are time-consuming, simulation enables to virtually experiment proposals with easy control of changing variables and parameters, allowing to gain good insight into the analysis and outcomes [52]. This has prompted us, since the beginning of the study, to use software simulation for better assessment of our approach.

We introduce, in this section, the technical details of our multi-cloud simulation tool (MCST) that has been developed on top of a well-known discrete event-based library called SimJava [63]. The MCST has been implemented entirely in Java using the Eclipse Modelling Framework (EMF)[1] and Java-8. It requires only Java Runtime Environment (JRE) 7 or later to be installed. The SimJava library supports the basic functionalities for building up a simulation system[2], such as the abstract creation of system entities (e.g., Cloud, Server) and links between them as well as the scheduling/processing of events using a simulation clock. In particular, MCST relies on four main classes of the SimJava library as follows:

- *eduni.simjava.Sim_entity*: A class to be extended for defining system entities each of which runs concurrently in its own thread. For instance, the declaration of our cloud-entity as a Java class begins with *public class CloudEntity extends Sim_entity* {..}. To practically simplify the simulation process, we break our cloud model (discussed in Section 3.1.1) into just three entities: *Source*, *Server* and *Cloud*. These top level entities are sufficient to simulate the behaviours of the proposed system model. Thus, we abstract away the unnecessary low-level entities that are related to the cloud infrastructures in general (e.g., VMs or services), but insignificant in the context of this thesis.
- *eduni.simjava.Sim_port*: A class for defining ports upon which two *Sim_entity* can be connected.

[1]Eclipse modelling Project http://www.eclipse.org/emf/, (2017).
[2]Simjava documentation is available at `http://www.icsa.inf.ed.ac.uk/research/groups/hase/simjava/`

- *eduni.simjava.Sim_event*: A class for defining events to be sent between entities.

- *eduni.simjava.Sim_system* : the kernel class, i.e., responsible for controlling entities, queueing, events, time, and report generation. It offers many runtime functionalities, essentially *sim_clock()* to get the current running simulation time, *run()* to launch the simulation, and *running()* to check if the simulation is still up.

In the following sections, we give more technical discussions regarding the design and implementation, configurations, and execution flows of MCST.

3.2.1 Structural Model and Implementation

A structural model of the MCST classes, represented using UML class diagram notation, is shown in Figure 3.3. It consists of four packages (*jobTracer, models, scheduling and simulation*) and 26 classes, see the API-documentation[3] of the public methods generated by Javadoc. These packages define the main components of the tool, explained in some detail as follows:

- *jobTracer*: This package consists of classes that are responsible for (1) parsing parallel workload logs to generate HPC-applications using *App* and *Task* classes. (2) determining the path of the critical tasks in the application in order to set precisely the time-frame to each task including the deadline.

- *models*: It consists of classes for defining the structure of our HPC-application as well as multi-cloud system. The latter structure appears as a hierarchy, organised by the class *Sky* at the top to the virtual CPU

[3]The API documentation of MCST is available at `http://www.cs.le.ac.uk/people/ayya1/MCSTdoc/index.html`

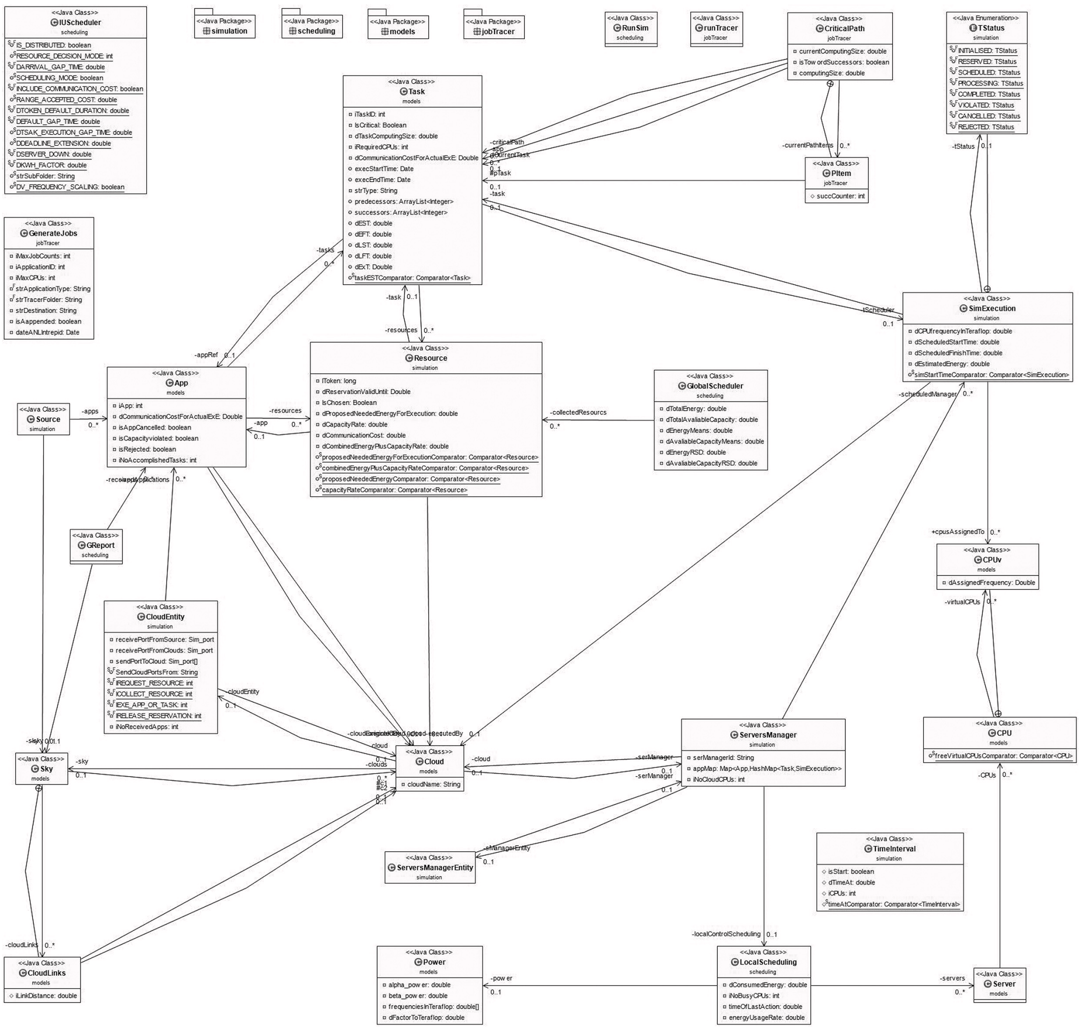


FIGURE 3.3: Class diagram of the MCST main classes

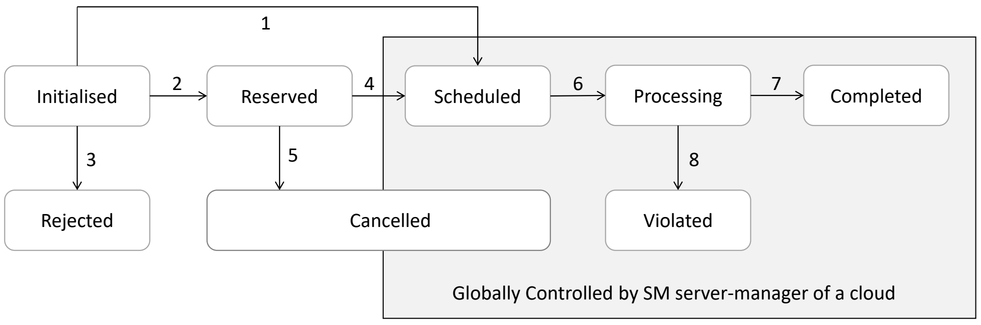


FIGURE 3.4: Task's states and defined transitions in our application model

class *CPUv* at the bottom. As a side note, the class *Sky* does not make our cloud model centralised, but instead, it just defines the global network between clouds. It implements our solution for computing the cost of data transmission between clouds. For modelling HPC-applications, the class *App* has an attribute container typed over class *Task*. Here, each *Task* has predecessors and successors lists for defining task dependency. The behaviour of a task is structurally defined in the class *TStatus*, which is expressed by 8 possible states to be changed during runtime. To illustrate more, Figure 3.4 describes the allowed transitions between these 8 states as the initial state must begin with *Initialised*, controlled by the original cloud o_{app}. The transitions to the states (*Reserved, Rejected*) occur during the global scheduling process (i.e., between participating clouds), while the rest (*Scheduled, Processing, Completed, Violated*) occur during the local scheduling process that is managed by the chosen (or executor) cloud. The transition to state *Cancelled* can happen in both the global (i.e., once a decision on which cloud is going to execute the submitted application is made, the unchosen clouds by the global scheduler will receive a request to cancel their reserved resources) and the local (i.e., once a token reservation period becomes invalid, occurs with e.g. a delay of the arrival datasets) scheduling process.

- *scheduling*: The classes in the scheduling package define simulation configurations (shown in Table 3.2), scheduling polices and their decision strategies, introduced in Section 4.2.
- *simulation*: This package consists of classes responsible for (1) implementing simulation entities using SimJava [63], and for (2) the generation of logs and statistical reports.

Property	**Description**
IS_DISTRIBUTED	A boolean property to specify the way of sending applications from *source* entity (representing e.g. end-users) to the clouds. If it is *true*, the applications will be submitted to all clouds in the multi-cloud system equally one-by-one. The *false* case means that all applications will be sent randomly to only a single cloud from the multi-cloud system.
RESOURCE_DECISION_ MODE	MCST supports two modes of deciding the best available resources offered by clouds during scheduling process in terms of reducing energy consumption. These modes are: [$LowestEnergyMode = 1$] or [$CombinationMode = 2$]. The latter mode is also motivated by minimising application rejections, which combines dynamically estimated energy usages along with resource occupation rates, introduced in Section 4.2.
DARRIVAL_GAP_TIME	This is to set the default waiting gap time (or delay) before the arrivals of submitted applications to participating clouds.
SCHEDULING_MODE	MCST supports two scheduling modes that are: *[BestEffortBased=false] or [ReservationBased=true]*. We discuss these two modes in detail in Chapters 4 and 5 respectively.
INCLUDE_COMMUNIC ATION_COST	A boolean property to either include (or exclude) energy cost that is consumed by data transmission between clouds.
RANGE_ACCEPTED_ COST	This is a preference factor that can be chosen as any fraction in the range [0,1) for determining the acceptable energy costs. It will be taken into account if the simulation is running under [$Lowestenergymode$].
DTOKEN_DEFAULT_DUR ATION	Default setting for the validity time of a given token reservation, such that the reserved resources will be released automatically if the token-ID becomes invalid.
DTASK_EXECUTION_ GAP_TIME	Default delay time to be set for assigning and releasing tasks to/from CPUs.
SPEEDOF_NETWORK_ SIGNAL	The assumed speed of the network signal in km/s.
DDEADLINE_EXTENSION	A preference percentage for extending the deadline of the application.
DSERVER_DOWN	A preference percentage for simulating sudden unavailable resources, for evaluating the reliability of the proposed scheduling algorithms.
DV_FREQUENCY_SCAL ING	A boolean property to either enable (or disable) dynamic voltage and frequency scaling (DVFS) feature.

TABLE 3.2: The main configurations and options for managing the process of MCST, defined in the class *IUScheduler*

3.2.2 Simulation Stages and Execution Flows

In order to use MCST one would need to install the tool packages into the workspace and then include them in the Java classpath of the application project. Then, the only required coding is to make three main changes in the static class *RunSim* as follows:

- Setting the configurations according to Table 3.2 for the considered scenario.
- Initialising two object instances from class *Sky* (i.e., representing multi-cloud model) and class *App* (i.e., representing HPC-applications in a DAG structure).
- Compiling and running class *RunSim* as a Java application.

Specifically, the execution of MCST is composed of two main stages after setting the scheduling configuration, described in the Java code fragment in Listing 3.1. This code gives a brief example of how to get started with the main method to run MCST. As a preparation stage, MCST requires generating HPC-applications from real workload-logs [95, 51]. Here, actual tasks (or jobs) of the applications are executed in the HPC-environment which would mimic a cloud's operation. These actual tasks allow to exemplify requirements of real applications, including task's dependencies. As a side feature, MCST provides a generic interface for parsing different formats of workload-logs[4], including LLNL-Thunder, LLNL-Atlas, LLNL-uBGL, and ANL-Intrepid.

The first stage of using MCST begins with initialising a set of HPC-applications (i.e, applications that have been generated from logs) that are assumed to be submitted for execution as well as declaring the characteristics of a set of cloud instances, including the communication's specifications between them, see *lines 23-30* in Listing 3.1. This stage enables evaluating each application as well as cloud instance with regard to the consumed energy for computational and I/O

[4]Logs of Real Parallel Workloads from Production Systems: `http://www.cs.huji.ac.il/labs/parallel/workload/logs.html`

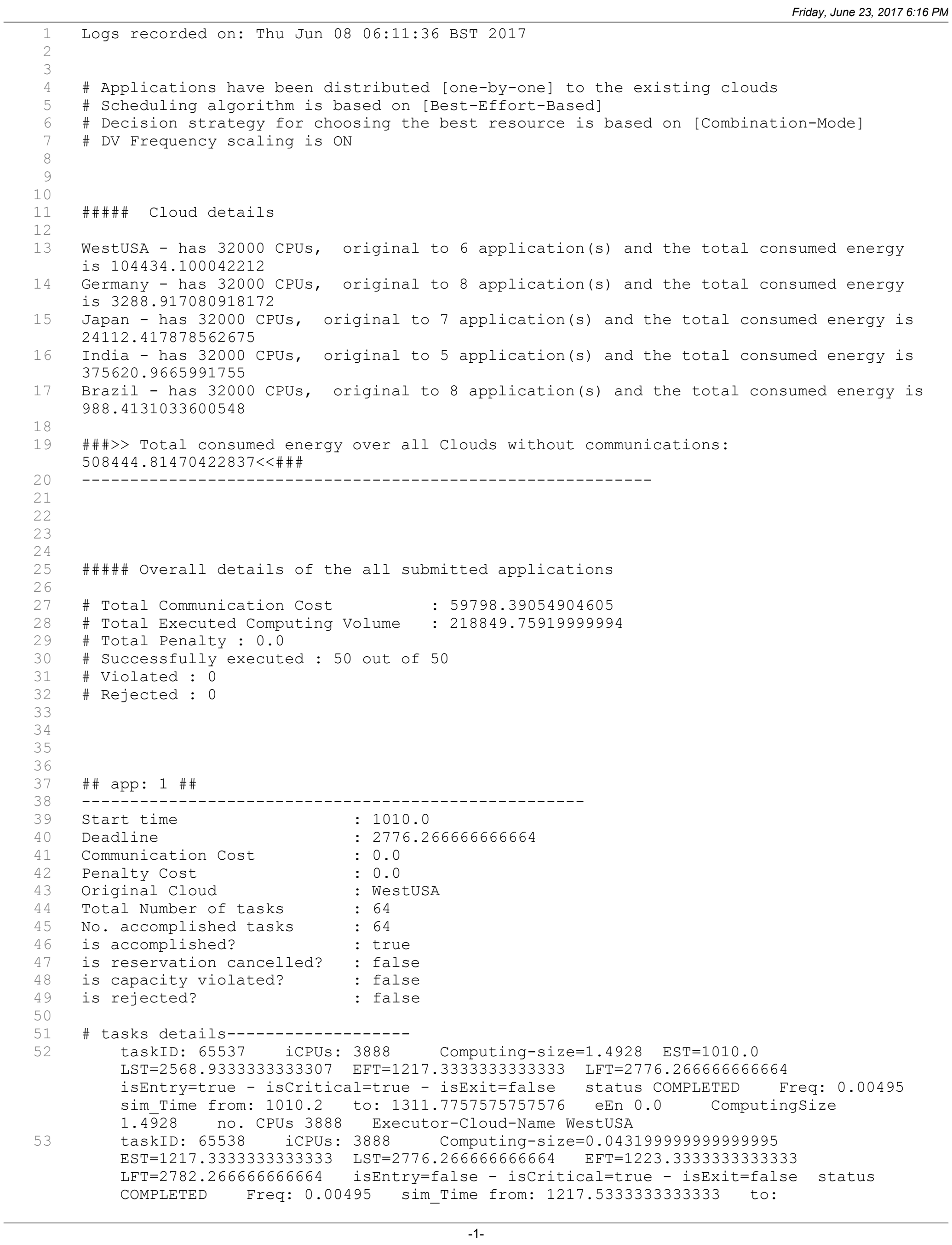
[5]`http://www.cs.le.ac.uk/people/ayya1/MCST/index.html`

```
 1 package scheduling;
 2 ...
 3 public class RunSim {
 4 
 5         private void mainSim(
 6                         String strOutFolder,
 7                         String strApplications,
 8                         int iNumberOfApplications,
 9                         int iNumberOfTasks,
10                         int iDecisionMode,
11                         double dPreferenceRange,
12                         double dTaskExEGap,
13                         double dDeadlineExt,
14                         boolean isDV_Freqency_Scaling){
15 
16                 /*
17                  * Set configurations */
18                 IUScheduler.RESOURCE_DECISION_MODE = iDecisionMode;
19                 IUScheduler.DDEADLINE_EXTENSION =dDeadlineExt;
20                 IUScheduler.DV_FREQUENCY_SCALING=isDV_Freqency_Scaling;
21                 ...
22 
23                 /*
24                  *  Initialise Sky object and applications from job tracers */
25                 Sky sky = new Sky();
26                 sky.InitialiseDefaultInstances();
27 
28                 ArrayList<App> apps =
29                         new GenerateJobs(iNumberOfTasks, strApplications).
30                         loadApplications(iNumberOfApplications);
31 
32                 /*
33                  *  Initialise Sim_system and main entities */
34                 Sim_system.initialise();
35                 new Source(sky, apps);
36 
37                 /*
38                  *  Run Sim and generate reports */
39                 Sim_system.set_trace_detail(true, true, true);
40                 Sim_system.run();
41                 new GReport(apps, sky);
42                 new TracingPerformance().toExcel(sky.getClouds());
43 }       }
```

LISTING 3.1: The main class used for running MCST tool

data transmission activities (i.e., how much energy one could reduce by utilising all available resources from all clouds).

The second stage (*lines 32-42*) focuses on report generation for analysis after launching MCST, see Figure 3.5 as an example of a generated report. Here, MCST optionally allows to generate specific logs for tracing the simulation's execution and getting better insights into the results. For example, it enables printing out the run time states of all clouds, describing utilised resources, when a specific application is being rejected or failed to execute.



```
Friday, June 23, 2017 6:16 PM
Logs recorded on: Thu Jun 08 06:11:36 BST 2017


# Applications have been distributed [one-by-one] to the existing clouds
# Scheduling algorithm is based on [Best-Effort-Based]
# Decision strategy for choosing the best resource is based on [Combination-Mode]
# DV Frequency scaling is ON



#####  Cloud details

WestUSA - has 32000 CPUs,  original to 6 application(s) and the total consumed energy
is 104434.100042212
Germany - has 32000 CPUs,  original to 8 application(s) and the total consumed energy
is 3288.917080918172
Japan - has 32000 CPUs,  original to 7 application(s) and the total consumed energy is
24112.417878562675
India - has 32000 CPUs,  original to 5 application(s) and the total consumed energy is
375620.9665991755
Brazil - has 32000 CPUs,  original to 8 application(s) and the total consumed energy is
988.4131033600548

###>> Total consumed energy over all Clouds without communications:
508444.81470422837<<###
------------------------------------------------------------




##### Overall details of the all submitted applications

# Total Communication Cost          : 59798.39054904605
# Total Executed Computing Volume   : 218849.75919999994
# Total Penalty : 0.0
# Successfully executed : 50 out of 50
# Violated : 0
# Rejected : 0




## app: 1 ##
---------------------------------------------------
Start time                  : 1010.0
Deadline                    : 2776.266666666664
Communication Cost          : 0.0
Penalty Cost                : 0.0
Original Cloud              : WestUSA
Total Number of tasks       : 64
No. accomplished tasks      : 64
is accomplished?            : true
is reservation cancelled?   : false
is capacity violated?       : false
is rejected?                : false

# tasks details-------------------
    taskID: 65537    iCPUs: 3888     Computing-size=1.4928  EST=1010.0
    LST=2568.9333333333307  EFT=1217.3333333333333  LFT=2776.266666666664
    isEntry=true - isCritical=true - isExit=false   status COMPLETED    Freq: 0.00495
    sim_Time from: 1010.2    to: 1311.7757575757576   eEn 0.0     ComputingSize
    1.4928    no. CPUs 3888   Executor-Cloud-Name WestUSA
    taskID: 65538    iCPUs: 3888     Computing-size=0.043199999999999995
    EST=1217.3333333333333  LST=2776.266666666664   EFT=1223.3333333333333
    LFT=2782.266666666664   isEntry=false - isCritical=true - isExit=false  status
    COMPLETED    Freq: 0.00495   sim_Time from: 1217.5333333333333   to:

-1-
```

Figure 3.5: An example of a statistical report generated by our tool (MCST). It mainly shows the energy consumptions by each cloud after scheduling and executing 50 HPC-applications over 5 distributed clouds. For more details, please see similar generated reports which have been made publicly available[5].

3.3 Comparison to Existing Tools

The existing simulation techniques can be broadly classified into either continuous or discrete-event simulation. We follow the latter, in a sense, as our MCST does not simulate any event continuously (e.g., no time loop for assigning a task to CPUs for execution) but simulates scheduling all events to run once at a specific time. In this section, we limit the discussion to only event-based simulation frameworks, designed for studying cloud computing systems in general. The closely related (non-commercial) tools we are aware of are GridSim [30], CloudSim [34], NetworkCloudSim [52], WorkflowSim [39], GreenCloud [79], and DynamicCloudSim [28]. Apart from GreenCloud [79], the core of all these tools, including ours, is SimJava [63], a basic discrete event simulation tool. Figure 3.6 describes the sequential extensions between these tools, inherited originally from SimJava [63] as a kernel-based framework.

Cloud simulators, to leverage experiments on medium and large scaled distributed environments, have been examined with different objectives and policies. They essentially allow to elaborate complex scenarios by concentrating on a certain component under different conditions. The historical beginning of the available tools was with grid simulators, such as GridSim [30] and SimGrid [37], for modelling and scheduling scientific applications over heterogeneous grid resources to address and evaluate performance problems. Following this, some of the grid simulation frameworks have been extended to support the requirements of cloud environments, including the model of virtualised resources and/or multi-layer services (e.g., IaaS). For example, the first version of CloudSim [34], which is one of the popular cloud computing frameworks, was an extension of GridSim [30].

The well-known CloudSim [34] is a self-contained toolkit that supports some basic features for modelling resource virtualisation (e.g., virtual machines and

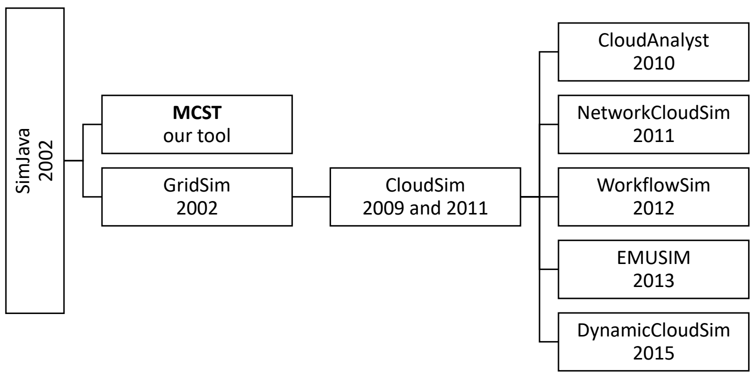


FIGURE 3.6: The sequential extensions between the related simulation tools to our MCST, that are inherited originally from SimJava [63]

CPUs) in data-centrers, allowing to implement/analyse scheduling policies for allocating tasks to virtual machines. It has been widely extended by many recent cloud simulators, including CloudAnalyst [115] and EMUSIM [33]. Nevertheless, the extensions that are most relevant to this thesis are: NetworkCloudSim [52], WorkflowSim [39], and DynamicCloudSim [28].

Helping simulation tools like CloudAnalyst [115] and EMUSIM [33] are designed to analyse different patterns of requests for cloud applications, containing information such as user and cloud locations, and capacity of cloud resources. The key idea behind such simulators is to assist users to accurately model their applications and to roughly estimate the performance/cost of their applications in a cloud provider. More specifically, EMUSIM as an example, derives information from application behaviours using emulation techniques to generate a model, describing the expected performance in a cloud provider. This descriptive model is then used to accurately specify application requirements as a simulation model.

For simulating and observing specified aspects of cloud behaviours, our tool MCST has in common with NetworkCloudSim [52], WorkflowSim [39], DynamicCloudSim [28], and particularly GreenCloud [79] the capability of scheduling applications over cloud resources to measure e.g., performance and/or energy consumption. However, each one of these simulators considers different

Feature	**MCST**	**A**	**B**	**C**	**D**	**E**	**F**
Parser for real archive workflow logs	✓				✓		✓
* modelling of heterogeneous multi-cloud system	✓		◇				
* modelling of complex HPC applications (dependent tasks)	✓	✓	◇	✓	✓		✓
* Energy model for calculating the cost of application computation	✓		◇			✓	
* Energy model for calculating the cost of data transmission	✓					✓	
* Energy saving based on DVFS	✓		◇			✓	
* Implementation of global schedulers over distributed multi-clouds based on token-reservation and best-effort modes	✓						
* Combination of conflicting objectives while scheduling	✓						
* modelling of application violation	✓				✓		✓
* modelling of application rejection	✓						
Supporting task execution, and scheduling algorithms, e.g., Round-Robin, MinMin [24], MaxMin [26], or any other heuristics	✓	✓	✓	✓	✓	✓	✓
Energy saving based on dynamic shutdown server						✓	
Precise models for local network in data-centres				✓		✓	
Support for task clustering					✓		
Random assignment of new VM to a host							✓
Dynamic changes of VM performance	✓						✓
Performance properties (MIPS, bandwidth or memory)			✓		✓	✓	✓
Performance property (file I/O)							✓

* Required feature in our approach.
◇ Feature is not fully implemented, but tool provides abstract classes to be extended.
A GridSim [30]
B CloudSim [34]
C NetworkCloudSim [52]
D WorkflowSim [39]
E GreenCloud [79]
F DynamicCloudSim [28]

Table 3.3: Comparison between the closely related simulation tools with our MCST

problem scenarios, aiming to address and achieve different objectives. For instance, the focus of NetworkCloudSim [52] and WorkflowSim [39] is ultimately on enhancing the measure of cloud performance by simulating a precise local network of data-centres and more realistic cloud applications, respectively. DynamicCloudSim [28] also share with them the enhancement of measuring cloud performance by modelling more critical factors, such as the cost of (1) file I/O, (2) dynamic changes to the performance of VMs, and (3) resource failures during task execution. In contrast, the scheduling approach presented in GreenCloud [79] focuses on energy optimisation for both servers and network activities within a single cloud-infrastructure. This, to some extent, is close to one of our objectives. In spite of it, GreenCloud [79] does not support the model of heterogeneous multi-cloud systems nor of complex HPC-applications. It is an extension of the NS2 network simulator, implemented in C++, and one of its drawbacks is that it can handle only small cloud resources, i.e., it has an exponential time complexity when simulating large data-centres.

In Table 3.3, we give a specific comparison between the relevant tools and our MCST. The features mentioned in this table that are supported by our tool are discussed throughout the thesis, e.g. modelling the energy formula and implementing the global schedulers are discussed in Chapters 4 and 5. However, the purpose of showing this table here is to make it clear why MCST has not been developed upon more recent tools than SimJava [63], particularly GridSim [30] or GreenCloud [79]. In a time-restricted experiment, these tools are difficult to use effectively without a major effort in terms of customising and/or modifying such relevant tools correctly. The main requirements to simulate our scheduling approach are not completely supported by these tools, see the rows that begin with (*) in Table 3.3. None of the mentioned tools straightforwardly supports our desired features, essentially modelling multi-cloud entities in a heterogeneous environment, such that clouds are owned by different vendors and each cloud has different characteristics of data-centres. Hence, we have concluded that integrating our tool with a fine-grained basic tool like simJava [63] is eventually an adequate solution to precisely evaluate our approach in less time and effort.

3.4 Summary

In this chapter, we have presented the Multi Cloud Simulation Tool (MCST) as a prototype developed to evaluate our approach. The MCST is entirely implemented in Java, and it provides an environment to abstractly represent distributed multi-cloud systems as well as HPC-applications to analyse the scheduling process of tasks and/or HPC-applications for energy efficiency. In particular, our tool provides the following main functionalities:

- parsing and loading applications from logs of real large-scale systems [95, 51];
- declaration of the characteristics of clouds and the specification of their communications; and
- generation of different reports for analysis.

Additionally, we have explained that none of the relevant tools straightforwardly supports our desired features, essentially defining our multi-cloud system, which results in the need of developing our own simulation for better and more precise evaluation. To estimate the amount of energy that could be efficiently reduced while scheduling HPC-applications over distributed multi-clouds, we introduce in the next chapter energy aware scheduling algorithms that rely on the best-effort scheduling style.

Chapter 4

Energy Optimisation with Best-Effort Scheduling Mode

Conforming to the main orientations of green cloud computing for the necessity of increasing energy efficiency to address some environmental problems (e.g., global warming), we investigate the problem of scheduling HPC-applications over a set of clouds, participating in a federated approach (cf. Section 1.1.2). This scheduling follows the best-effort style (i.e., non-advance-reservation style) when allocating tasks to clouds and/or machines for execution. It exploits the possibility of sharing cloud resources from different vendors as well as the elasticity of dynamically adjusting voltage and frequency of processors (DVFS) (cf. Section 2.2.1) for energy optimisation.

This chapter presents a deadline-based scheduling method for optimising the energy consumption when executing dependent HPC tasks by a decentralised multi-cloud system. The optimisation involves a non-trivial amount of energy consumed by the execution of tasks and the transmission of their datasets, if applicable, between clouds. A preliminary version of this method that focuses on only optimising the computing-energy consumption has been published in [6]. Along with energy efficiency as the major objective of this thesis, the proposed scheduling method supports maximising the resource reliability by the avoidance of rejected/violated application cases.

The chapter is structured as follows. In Section 4.1, we present the problem under consideration in terms of the multi-cloud system model, scheduling model and energy model followed by the problem formulation. We then introduce, in Section 4.2, two different decision strategies for choosing the best available resources, that are offered by a set of clouds. These are *Preference Rate* to optimise energy consumption based on setting an upper limit for the allowed energy consumption, and *Combination Rate* to address conflicting objectives using a statistical approach. Section 4.3 presents our proposed energy aware scheduling algorithms as two of the prime contributions of this thesis: EGSBE for scheduling tasks among all participating clouds, and ELS for allocating each task to suitable machines in a local datacenter. After that, we evaluate the proposed approach using our simulation (MCST) in Section 4.4. Finally, the chapter ends with a brief summary in Section 4.5.

4.1 Scheduling and Energy Models

In Section 3.1, we have introduced the proposed system model, including the considered cloud as well as application structures, for our approach. The model suggests sharing the execution of HPC-applications by distributed and decentralised clouds that participate in a federated approach. To make such participation serviceable, essentially for energy optimisation, we present, in the next section, a scheduling framework based on the best-effort mode. The framework consists of two dynamic schedulers: *global* for assigning application tasks to clouds, and *local* for mapping tasks to CPUs. Both schedulers are constrained by application/task deadlines while optimising energy consumption. Considering energy models, we propose two formulas for calculating energy consumptions:

one for estimating the execution cost of a task in a cloud, and the second for estimating the cost of transmitting a dataset between clouds, see Sections 4.1.2 and 4.1.3, respectively. The main objective of the proposed best-effort scheduling approach is explained formally in Section 4.1.4.

4.1.1 Scheduling Framework

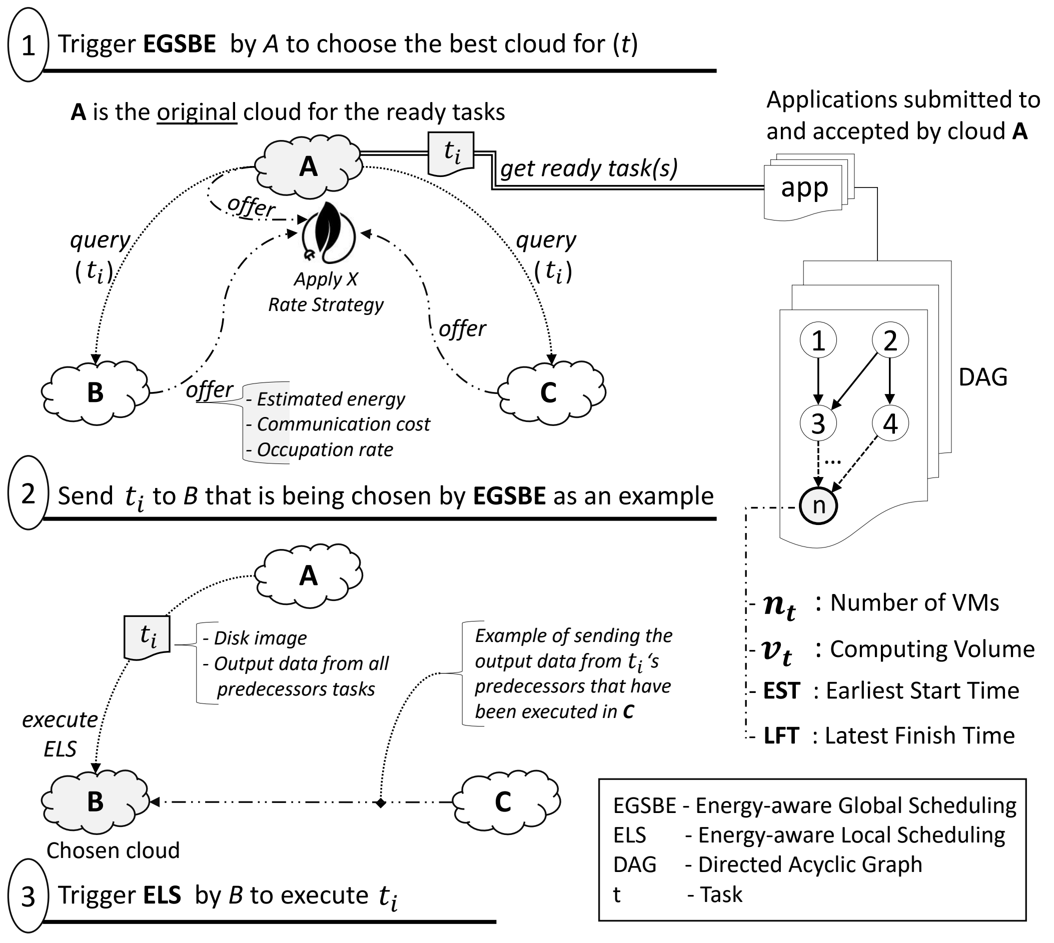
As illustrated previously in Figure 3.1, a cloud site has a *global scheduler*, a *local scheduler*, and a *resource controller* that are managed by the cloud's manager server. Here, each manager server has mainly four queues for controlling tasks based on the task's status (described in Figure 3.4) as follows:

- *Unprepared tasks queue*: This is the first stop of all tasks, where the status of these tasks is *Initialised*.
- *Ready tasks queue*: It contains a list of tasks that are waiting for replies of delegation requests from other clouds until the global scheduling decision is made. The status of all of these tasks is also *Initialised*. However, these tasks are considered ready as the executions of all of their predecessor tasks have been successfully performed.
- *Delegated tasks queue*: This queue contains a list of the globally scheduled tasks, but the executer cloud (that has received the disk image of such delegated tasks as a kind of notification for accepting the offer) is waiting to receive the outputs of all the predecessors of the delegated task from other clouds. The status of these tasks, in this queue, is *Scheduled*.
- *Task scheduling queue*: It hosts all tasks that have been accepted, and are under processing by local resources. The status of these tasks is *Processing*.

Consider the following scenario to illustrate the role of the main components of each cloud: The manager server ms_1 receives a submitted application from a user before partitioning it into tasks while maintaining the dependencies among them. The *global scheduler* relies on one of the suggested scheduling strategies (discussed later in Section 4.2) that analyses the information about local and remote resources as well as the task's requirements to decide to which cloud each task will be allocated. The information about local resources is obtained directly through the local *resource controller*, while the information about remote resources is obtained by communicating with other manager servers $ms_2, \cdots, ms_n$, where each of them is in turn related to a *resource controller*. Then, the *local scheduler* of the chosen cloud decides which servers/processors are allocated to each task based on the utilised scheduling policy and considering the requirements of the task. In a little more detail, the process of the *global* and *local* schedulers for allocating tasks is illustrated in three steps in a self-contained figure, see Figure 4.1. With regard to the communication among multiple clouds, we adopt a pull and push (P&P) model [64] as the resource monitoring mechanism (cf. Section 2.4.1). The broadcasting of the delegation request messages from the original cloud to all participating clouds represents the pull operation, and the notification messages from the executer cloud to the original cloud (either of task rejection, failure, or successful execution) represent the push operation.

4.1.2 Energy Formula for Application Execution

In our energy-modelling context, specifically for execution activities, the focus is on the processor only as it is one of the main critical components of the entire cloud system in terms of energy consumption [55]. All of the other important components (e.g., memory or disk drives) are beyond the scope of this thesis. In Section 2.2.2, we have explained the basic formula for computing the energy



1 Trigger EGSBE by A to choose the best cloud for (t)
A is the original cloud for the ready tasks
Applications submitted to and accepted by cloud A
t_i
get ready task(s)
app
A
offer
query (t_i)
query (t_i)
Apply X Rate Strategy
offer
B
offer
C
- Estimated energy
- Communication cost
- Occupation rate
DAG
1
2
3
4
n
2 Send t_i to B that is being chosen by EGSBE as an example
A
t_i
- Disk image
- Output data from all predecessors tasks
Example of sending the output data from t_i's predecessors that have been executed in C
- n_t : Number of VMs
- v_t : Computing Volume
- EST : Earliest Start Time
- LFT : Latest Finish Time
execute ELS
B
C
Chosen cloud
3 Trigger ELS by B to execute t_i
EGSBE - Energy-aware Global Scheduling
ELS - Energy-aware Local Scheduling
DAG - Directed Acyclic Graph
t - Task

FIGURE 4.1: Overview of the proposed best-effort approach, described in three steps

consumption by a single-core processor that supports DVFS. In this thesis, we adopt an extension of this traditional energy formula to support multi-core processors, as follows.

Assume that p is a multi-core processor, supporting DVFS, and CO_p is the set of available cores in p, where $|CO_p| > 0$. If all cores CO_p run at the same frequency f, the traditional formula (discussed in Section 2.2.2) can be modified to:

$$E_p = \beta + x(\alpha f^3) \tag{4.1}$$

where x is the number of active cores in p. Here, x has no impact on the static energy of the actual physical processor (represented by β), but it affects directly the dynamic energy [77]. In our energy model, we assume a little more complex case in which the active cores of a processor are allowed to run simultaneously at different frequencies, as in [82, 48]. In this case the system has a set of identical clocks, one for each core, to allow varying performance among the cores. This might occur if each core is assigned to different tasks that need to be executed at different frequency. If core $co \in CO_p$ runs at frequency f_{co}, the energy metric for the multi-core processor p becomes: $E_p = \beta + \sum_{co \in CO_p} \alpha f_{co}^3$. Thus, our proposed formula to compute the total energy consumption E_{c_j} by a set of servers S in the cloud site c_j is expressed as follows:

$$E_{c_j} = \sum_{s \in S} \Big(\sum_{p \in P(s)} \big((\beta_j + \sum_{co \in CO_p} \alpha_j f_{co}^3) D_p \big) \Big) \tag{4.2}$$

where $P(s)$ is the set of processors of server s, D_p denotes the active time of processor p, and f_{co} denotes the frequency level at which core $co \in CO_p$ for some processor p runs.

Furthermore, we calculate the estimated energy consumption by a particular task t as follows. Assume that n_t is the number of VMs assigned to t such that each VM occupies one core, and these VMs run at frequency f. Then, the total energy consumption by task t can be expressed as:

$$E_t = (\beta + N(\alpha f^3)) \times ceil(\frac{n_t}{N}) \times e_t \tag{4.3}$$

where $e_t = \frac{v_t}{f}$ is the execution time of task t at frequency f, N is the number of cores per processor, and $ceil(\frac{n_t}{N})$ represents the estimated number of physical processors that are assigned to t.

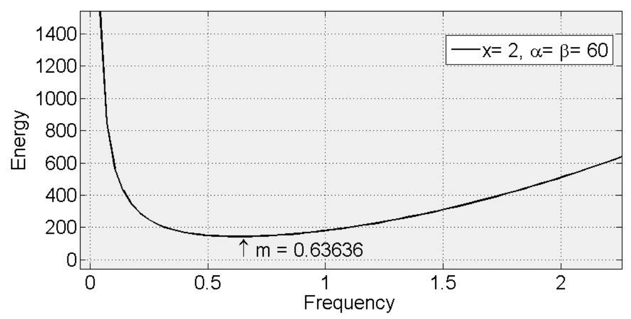


FIGURE 4.2: Energy consumption vs. frequency

However, due to the convexity of the mentioned energy metric, not all lower frequency levels are useful for minimising the energy consumption. We can define a useless frequency f as a frequency at which the processor, when executing a fixed volume of computation, always dissipates an amount of energy that is larger than the amount of energy dissipated at the frequency m that minimises the amount of energy, and if f belongs to the interval $[f_{min}, m)$. Figure 4.2 shows an example of the energy consumption per unit of computation of a dual-core processor with $\alpha = \beta = 60$ and a frequency range $[0.17, 2.5]$, where the interval of the useless frequencies is $[0.17, m)$. Despite the fact that these useless frequencies may have an instantaneous energy consumption that is lower than that of higher frequencies, they always need more energy in total for executing a task than some higher frequency that finishes the task sooner.

Thus, we eliminate useless frequencies as follows. For each frequency f, we compute the amount of energy consumed by a processor for one unit of computation on each core, which is given by $Energy = \frac{\beta+(x*\alpha f^3)}{f}$ for a processor with x cores. We then remove all frequencies that are smaller than the frequency that minimises this energy. This elimination, where applicable, helps to reduce the computation time of the scheduling method that selects a suitable frequency for executing a task.

4.1.3 Energy Formula for Data Transmission

The computation of transmission energy depends mainly on the cost of wired connections through which the dataset of a given size is transmitted. The link cost combines the total consumption of the Internet nodes and cooling, the transmission lines, and amplifiers [44]. When a non-original cloud (i.e., not o_{app_m}) executes an application app_m, the datasets, including the input and the disk image, need to be transmitted to the executer cloud. Once the execution of app_m is completed, the output will also need to be transmitted back to the o_{app_m}.

Estimating the energy consumption of data transmissions through the Internet is notoriously difficult, and available estimates vary by several orders of magnitude [44, 45]. We adopt an estimate of 0.2 kWh for the transmission of 1 GB as this value lies in the middle region of the range of reported estimates. Furthermore, to account for the effect that transmissions over longer distances are likely to require more hops and thus more energy, we assume that the energy consumption of a data transmission also depends linearly on the distance over which the data is being transmitted. We make the assumption that a typical transmission to which the rate of $\mu = 0.2kWh/GB$ applies is a national transmission over a distance of 500 km, so that the energy cost of a transmission over a distance D can be obtained by multiplication with the factor $D/500\,km$. In general, if a more accurate estimation of transmission energy costs is available for a given scenario, it can be incorporated into our simulation MCST in a straightforward way.

We assume that all datasets of an application will be sent through the same link. Given a set A of delegated applications whose datasets need to be sent from o_{app_m} to the executor cloud, we estimate transmission energy T_{c_j} as:

$$T_{c_j} = \sum_{app_m \in A} (\mu \times dataSize_{app_m} \times \frac{linkLG_{app_m}}{500\ km}) \tag{4.4}$$

where $dataSize_{app_m}$ denotes the total size of the disk image, the input and output, and $linkLG_{app_m}$ expresses the link-length used for the transmission.

4.1.4 Problem Formulation

We consider a set of applications $\mathcal{A}$, submitted over time to different specified cloud sites in a multi-cloud system, where $\mathcal{A} = \{app_1, \ldots, app_L\}, L \in \mathbb{N}$. Cloud c_j may receive y applications, where $0 \leq y \leq L$. The submission of app_m is unknown beforehand. Each cloud can accept a received app_m if the deadline can be met, or reject it otherwise. If the app_m gets accepted for scheduling, it will be either executed successfully or violated. Our objective is to primarily optimise the total energy consumption of all accepted applications in the entire multi-cloud system with the avoidance of rejections and application violation cases.

We attempt to minimise the overall energy usage E_{total} that includes (1) the computing energy usage E_{c_j}, (2) the dataset transmission energy cost T_{c_j} and (3) the penalty cost for rejecting/violating applications PN_{c_j} by cloud c_j. Here, the penalty cost PN for rejecting/violating an application app_m is $PN = \sum_{t \in app_m} E_t$, where E_t is calculated by equation (4.3) at the highest performance among all clouds. Thus, the objective function is as follows.

Minimise: $E_{total} = \sum_{c_j \in C}(E_{c_j} + T_{c_j} + PN_{c_j})$

Subject to:

(1) $endTime_{app_m} \leq DL_m \quad \forall\ app_m \in \mathcal{A}$ where app_m is accepted

(2) $f_{min_j} \leq f_{co} \leq f_{max_{c_j}} \quad \forall\ co$ in servers of $c_j \in C$

By this objective function, a scheduling policy that rejects all applications would have $\sum_{c_j \in C} E_{c_j} = \sum_{c_j \in C} T_{c_j} = 0$, but it gets a very high $\sum_{c_j \in C} PN_{c_j}$. The policy that executes all applications at the highest cloud performance would tend to have a very low penalty but a high $\sum_{c_j \in C} E_{c_j}$. The scheduling policy which will have a better objective value is the one that finds a good balance.

4.2 Schedule Decision Strategies

Consider a situation where we are given two lists of numerical values offered by a set of clouds, and the goal is to pick a cloud that optimises these values, taking into account the priority of sorting these two lists. In this research context, these two lists represent offers from clouds for scheduling and executing a particular submitted application, such that the first list collects the estimated energy consumptions, while the second list collects the occupation rates of all clouds' resources. To give more clarification, we exemplify these two lists with values offered by 5 clouds as illustrated in Figure 4.3. Here, picking a cloud by focusing on only a single list (e.g., a list that collects the estimated energy consumptions), and by finding the minimum value (e.g., *cloud B* that offers 1.0001 in Figure 4.3) may not lead to the optimal energy optimisation over time. Instead, taking cloud resources that are occupied by the other applications into consideration is very critical.

When the differences between the values of the estimated energy consumptions that are offered by clouds are negligible (e.g., the first three values in the first list in Figure 4.3), it makes obvious sense to select a set of clouds that offer similar minimum values, and then consider the rates of their resource occupations as the next priority. This is concisely described in Figure 4.3 as the differences between clouds (B, E, D) in the first list seem very minor, but become significant in clouds (A, C). This means picking any cloud from (B, E or D) would make

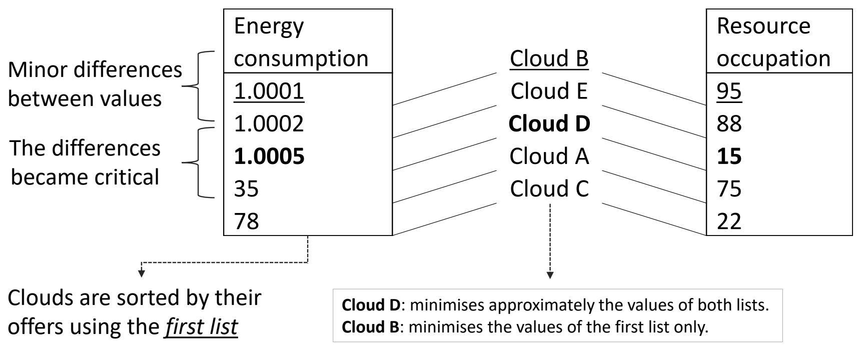

Minor differences between values
The differences became critical
Energy consumption
1.0001
1.0002
1.0005
35
78
Cloud B
Cloud E
Cloud D
Cloud A
Cloud C
Resource occupation
95
88
15
75
22
Clouds are sorted by their offers using the *first list*
Cloud D: minimises approximately the values of both lists.
Cloud B: minimises the values of the first list only.

Figure 4.3: An illustrative example explaining abstractly the form of the clouds' offers. It represents some assumed values, in two lists, offered by 5 clouds.

almost no difference in terms of reducing energy consumption. However, when considering resource reliability at the level of the whole multi-cloud system over time, *Cloud D* offers the best choice as it also minimises the occupation rate (e.g., see the value 15 in the second list), meaning that it has less chance of running out of resources.

In this thesis, we propose two decision strategies: preference and combination rate strategies for finding the best available resources, introduced in the next subsections. These strategies depend on some pre-defined configurations in our simulation (MCST), including *RESOURCE_DECISION_MODE* and *RANGE_ACCEPTED_COST*, which were explained briefly in Table 3.2.

4.2.1 Preference Rate Strategy (PRS)

Assume that the first and the second lists, shown in Figure 4.3, represent the offers from a set of clouds such that the first list collects the estimated energy value ($eEnergyValue$) and the second list collects the occupation rate ($occupRt$). Here, PRS minimises primarily the energy consumption based on a given preference factor Rt that can be chosen as any fraction in the range [0,1). This factor needs to be defined in MCST using *RANGE_ACCEPTED_COST*.

The idea is to determine the acceptable energy costs by assigning the minimum energy value $min(eEnergyValue)$ provided as the lower bound, while choosing $min(eEnergyValue) + (min(eEnergyValue) \times Rt)$ as the upper bound. Only clouds whose $eEnergyValue$ lies in this range are then considered, and the strategy then minimises $occupRt$ among their offers.

Intuitively, if the given factor Rt is 0, PRS would always choose the cloud that gives the minimum energy immediately without considering $occupRt$. Accordingly, the applications will be executed at the minimum offered energy consumption. To be more precise, consider the descriptive example in Figure 4.3, the PRS will choose *Cloud B* when setting $Rt = 0$. On the other hand it would choose *Cloud D* as a better decision when expanding the range of the acceptable $eEnergyValue$ by setting e.g. $Rt = 0.05$, such that the upper bound would be $1.0001 + (1.0001 \times 0.05) = 1.0501$, allowing to minimise $occupRt$ from the determined range (i.e., $1.0001, 1.0002$, and 1.0005).

4.2.2 Combination Rate Strategy (CRS)

Unlike PRS, CRS aims to simultaneously satisfy the minimisation of both the estimated $eEnergyValue$ and $occupRt$. The strategy is inspired by two statistical analysis concepts that are the standard deviation SD and the coefficient of variation RSD, which are commonly used as metrics for assessing the robustness of a schedule [86]. They effectively suit our approach as the SD of a set of values e.g. $eEnergyValue$ expresses how much the proposed energy values differ from their mean value, and this helps to understand the amount of their dispersion, while, the RSD gives us a ratio scale that allows us to compare two different (non-unified) distributions based on their dispersion. To avoid any ambiguity, let us consider that the SD of a set $Energy = \{en_1, \cdots, en_a\}$ can be calculated by (4.5):

$$SD = \sqrt{\frac{\sum_{en_i \in Energy} |en_i - \overline{Energy}|^2}{a}} \tag{4.5}$$

where $\overline{Energy}$ is the mean of the data set $Energy$, calculated by $\overline{Energy} = \frac{\sum_{en \in Energy} en}{a}$. The coefficient of variation RSD is the ratio of standard deviation to the mean $\overline{Energy}$, and can be expressed as: $RSD = SD/\overline{Energy}$.

To pick the best cloud based on this strategy, the CRS forms two lists *eEnergyList* and *occRtList* with all proposed $eEnergyValue$ and *occupRt*. Here, the RSD of each list represents the amount of dispersion between the elements such that low dispersion would refer to very similar offered values, which may make no difference when choosing any element. High dispersion, however, means that it is important to consider each element as there is a clear difference between the elements.

Having the RSD from *eEnergyList* and *occRtList*, it makes sense to use these lists to determine a weight for each objective in a weighted sum, such that:

- the first weight is $\frac{RSD(eEnergyValue)}{RSD(eEnergyValue)+RSD(occupRt)}$, and
- the second weight is $\frac{RSD(occupRt)}{RSD(eEnergyValue)+RSD(occupRt)}$.

We then compare the value of these two derived weights to determine the higher weight hw as well as the lower weight lw. Therewith, the defined hw will be given to the set of items that are highly dispersing (i.e., either list *eEnergyList* or *occRtList* that has a higher RSD), and accordingly the lower weight lw will be given to the other list. These weights (hw and lw) as well as the RSDs of both lists (*eEnergyList* and *occRtList*) are calculated dynamically each time CRS is applied, and after collecting cloud responses for scheduling each task.

Assuming that $eEnergyValue$ has a higher weight hw, the CRS will choose the cloud with minimum combined rate from the list: $\{(eEnergyWeight_1 * hw +$

	Energy			Resource occupation rate			
Cloud	eEnergyValue	Energy weight	Energy weight * 0.696058981	occupRt	Occupation weight	Occupation weight * 0.303941019	Combined energy & occupation rates
B	1.0001	0.008621492	0.006001067	95	0.322033898	0.097879311	0.103880378
E	1.0002	0.008622354	0.006001667	88	0.298305085	0.090667152	0.096668819
D	*1.0005*	0.008624941	0.006003467	*15*	0.050847458	0.015454628	***0.021458095***
A	35	0.301722057	0.210016348	75	0.254237288	0.07727314	0.287289488
C	78	0.672409156	0.468036432	22	0.074576271	0.022666788	0.49070322
Mean	23.20016			59			
Standard Deviations	33.98810262			37.74254893			
Coefficient of variation	1.46499432			0.639704219			
Given percentage	0.696058981			0.303941019			

Table 4.1: An example describing how our proposed combination rate strategy (CRS) can be applied.

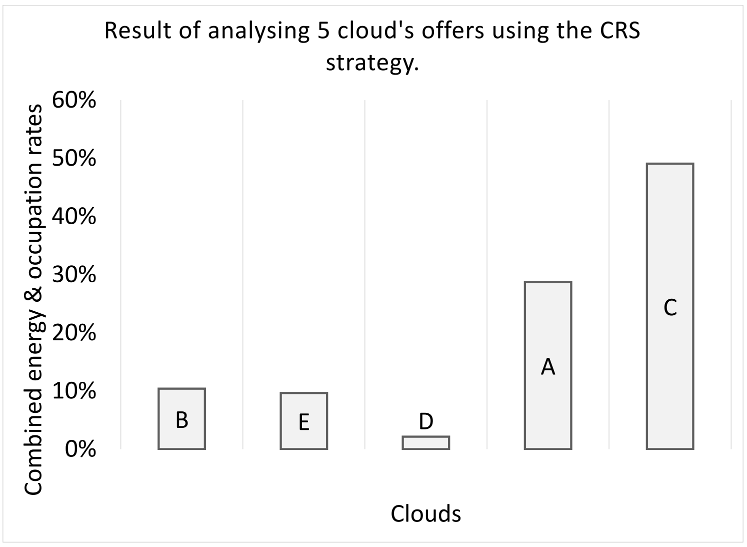


Figure 4.4: Applying the CRS strategy to the example presented in Figure 4.3, which results in the choice of *Cloud D* as the best option.

$OccupationWeight_1 * lw), \cdots, (eEnergyWeight_a * hw + OccupationWeight_a * lw)\}$, where $eEnergyWeight_i$ is $\frac{eEnergyValue_i}{eEnergyValue_1 + \cdots + eEnergyValue_a}$, $OccupationWeight_i$ is $\frac{occupRt_i}{occupRt_1 + \cdots + occupRt_a}$, and a is the number of received offers.

Table 4.1 explains explicitly how CRS can be applied based on the descriptive example shown in Figure 4.3, which results in the choice of *Cloud D* as the best option, visualised in Figure 4.4. More precisely, the second ($eEnergyValue$) and the fifth ($occupRt$) columns give the offer values from the 5 clouds, the same values shown in Figure 4.3, and also the *mean*, SD and RSD of these values.

From the calculated $RSDs$ of both columns (1.46499432 and 0.639704219 in Table 4.1), we define hw and lw as explained, and then assess the dispersions of the offer values, expressed by the ratio of SDs to the mean of the offer values. The column (i.e., either the $eEnergyValue$ or the *occupRt*) that has the higher RSD will take the hw, whereas the lw will be given to the other column. For example, in Table 4.1, the RSD of *occupRt* is 0.639704219 which is lower than the RSD of $eEnergyValue$ that is 1.46499432. Accordingly, the column $eEnergyValue$ has taken the hw that is 0.696058981, while the lw (0.303941019) is given to the column *occupRt*, see the last row. The last column combines the considered energy weights as well as the considered occupation weights based on the given hw and lw, such that the cloud that gives the minimum value in this column is chosen.

4.3 Scheduling Algorithms

Before presenting the proposed scheduling algorithms, we first explain the general scheduling process. It is dynamic, and follows the best-effort (non-advance-reservation) style. In this style, the scheduling decision is made depending on some instant information about all available resources in the entire multi-cloud system. The given offers have no guarantee for the resource's availability. The scheduling process starts when a submitted application is accepted. The acceptance decision is made through estimating the execution time of the whole application by calculating the total execution time of its critical tasks at the highest performance among all clouds. The execution time must be less than or equal to the submitted deadline DL_m, otherwise the application will be rejected. The deadline d_{t_i} for each task is calculated as a guide for the global scheduler as follows: $d_{t_i} = LFT_{t_i}$ where LFT_{t_i} is calculated using (3.2).

The task t_i of an accepted application is considered as a ready task for scheduling only if it has no predecessors (i.e., an entry task) or all its predecessors have been successfully executed. If t_i is ready, then the preparation for its scheduling happens by broadcasting the delegation requests that are sent by the manager-server of the original cloud to all other manager-servers in the multi-cloud system. There are two types of replies for a delegation request, as follows.

- A positive reply implies a delegation acceptance, to which a task bidding information is appended, i.e., the estimated energy consumption for both execution and data transmission as well as the occupation rate.
- A negative reply, however, means that the delegation request is rejected. It occurs if either the task's deadline cannot be met by any means or the available resources are not enough. Also, there is a possibility that a task may not receive any positive reply even from the local resource, which means that the violation of its deadline is inevitable and the whole application will be violated in turn.

Broadly, after assessing the possible scheduling decision, the task can be allocated either to local resources of o_{t_i} or to remote resources in another cloud. The output data of a successfully executed task is called the dataset and its volume is represented by $dSet_{t_i}$. The dataset is necessary for all successors of t_i to be ready for scheduling and execution.

There are particular time periods (delays) that occur throughout the scheduling time of an application before starting the execution of each task and after a successful execution. These are:

- The time required to transfer the disk image of task t_i from o_{t_i} to the chosen cloud, $dImage_{ti}^{transfer}$, where this time is equal to zero if o_{t_i} is the executer cloud.

- Inner delay time, $innDelay_{t_i}$, is the time taken to transfer $dImage_{t_i}$ simultaneously to a set of computing nodes that are allocated to t_i.
- Data set transfer time, $dSet_{t_i}^{transfer}$, is the time taken to transfer datasets from all of t_i's predecessors from one or more data-centres to another. This occurs if the scheduler allocated t_i to a different cloud from the one in which t_i's predecessors are executed.
- Delay time of notifying o_{t_i} about a successful execution of t_i. It is denoted by $endNotify_{t_i}$.
- The transfer time of the output data of a delegated exit task t_i, $exitOutput_{t_i}^{transfer}$, from the executer cloud to o_{t_i}.

We assume here that the datasets that are required to execute a given task can be transferred to its allocated cloud only after successfully transferring its disk image to that cloud. It is also worth mentioning that the energy transmission cost that is attached to each task for being delegated does not include the cost of a successful-execution notification which is negligible.

4.3.1 EGSBE: Energy-Aware Global Scheduling Based on the Best-Effort Mode

The *EGSBE* schedules each task of an application to the cloud that offers the best option, which minimises the energy while meeting the deadline, including the option of scheduling it on the local cloud resources. Specifically, it relies on information about the resource status, i.e., the occupation rate *occupRt* in addition to the estimated energy consumption $eEnergyValue$ that consists of the computation and communication energy consumption. The steps of obtaining the information from all clouds for scheduling task t are shown by Algorithm 1 in *lines 1-4*. The scheduler will allocate tasks to their original cloud o_t (if their deadlines can be satisfied) unless another cloud offers a lower amount of energy consumption. In *lines 5* and *6*, if the deadline of a task can be met by a cloud c_i, the cloud offer will be added to the set *offers*.

Algorithm 1 EGSBE
Energy-aware global scheduling with best-effort.

```
Inputs: $t$ is a ready task to be scheduled by $o_t$. Each $t$ has a computing volume
$v_t$ and deadline $d_t$
        $C = \{c_1, \cdots, c_k\}$, $k \in \mathbb{N}$ is a set of connected clouds
        $PRS$ (preference) and $CRS$ (combination) are rate strategies.
Outputs: A globally scheduled $t$ in $c_i$, the best cloud found.
Begin
 1: $offers := \emptyset$ is a set of offers provided from all $c_i$
 2: for each $ms_i$ cloud manager server $\in C$ do
 3:    $offer \leftarrow ms_i$.getOffer($t$)
 4:    $offer.eEndTime_{t(ms_i)} := offer.EAT_{t(ms_i)} + (v_t / offer.Frequency)$
 5:    if $offer.eEnergyValue > 0$ and $offer.eEndTime_{t(ms_i)} +$
       $possibleDelay_{t(ms_i)} \leq d_t$ then
 6:       $offers \leftarrow offers \cup \{ offer \}$
 7: if $offers == \emptyset$ then
 8:    set $t.app$.isRejected=true
 9: else if $offers.size == 1$ then
10:    set chosenCloud($t$,$offers$(1)), i.e., the available offer is one, and hence
       schedule task $t$ to it directly.
11: else
12:    $bestOffer \leftarrow$ applyDecisionStrategy ($offers$, $CRS$ (or $PRS$)). Note, the
       $CRS$ and $PRS$ rely on the two values provided from each $offer$ that are:
       $offer.eEnergyValue$ and $offer.occupRt$, discussed in Section 4.2.
13:    set chosenCloud ($t$, $bestOffer$)
End
```

The ability of meeting the deadline of task t is estimated for cloud ms_i by computing the summation of $eEndTime_{t(ms_i)}$ and $possibleDelay_{t(ms_i)}$, see *line 5.* The latter is the delay that is associated with each task for being scheduled to a specified cloud. It may consist of one or more particular periods of time, as illustrated earlier in this section. Specifically, the $possibleDelay_{(o_t)}$ of the original cloud includes only the $innDelay_{(o_t)}$, whereas the $possibleDelay_{t(ms_i)}$ of all non-original clouds includes $innDelay_{t(ms_i)}$ in addition to either $endNotify_{t(ms_i)}$ or $exitOutput^{transfer}_{t(ms_i)}$.

Then, in *lines 7-13*, there are two cases: scheduling t to the chosen cloud (i.e., *line 10* describes the case if only one valid option is available and *line 13* for more than one options) or rejecting it as there is no cloud that is able to meet its deadline. The latter implies that the deadline of the whole application cannot be met, and the application is rejected in turn, see *line 8*.

Algorithm 2 ELS
Energy-aware local scheduling.

```
Inputs: Task $t(n_t)$.
        $capacity(S_c)$ the capacity of servers in this cloud.
        The list of all servers.
Outputs: Allocating the required machines to $t$.
Begin
 1: $Rn := n_t$
 2: form the list of all active servers $activeServersList$
 3: sort the $activeServersList$ in ascending order of their free capacity.
 4: for each server $s \in activeServersList$ do
 5:    form the list of processors that have free capacity $CPUsList$
 6:    sort the $CPUsList$ in ascending order of their free capacity.
 7:    for each processor $p \in CPUsList$ do
 8:       allocate a number of VMs that fulfill $Rn$ if available, or equal to the
          number of free VMs otherwise.
 9:       reduce $Rn$ value by the number of allocated VMs.
10:       if $Rn = 0$ then
11:          break
12: if $Rn > 0$ then
13:    activate $ceil(Rn/capacity(S_c))$ idle servers.
14:    allocate the remaining number of required VMs that is equal to $Rn$.
15: start executing the task $t$.
End
```

4.3.2 ELS: Energy-Aware Local Scheduling Algorithm

For each task t in the application *app*, the ELS algorithm is responsible for assigning t to servers, processors, and cores that will execute it, according to the schedule computed by Algorithm 2. The ELS is triggered whenever there is an immediate arrival of a delegated task, or the execution of a task t_i is due to start. The *app* will be violated if t has failed to execute (e.g., by not getting enough resources at run time), leading to cancelling the execution/scheduling of all of its tasks.

The goal of Algorithm 2 is to choose the resources in a way that helps minimising the computing energy consumption. Thus, it initially tries to utilise as many active servers as possible, in *line 4*, so as to reduce the cost of activating idle servers. It also utilises the active processors that have free virtual capacity in order to minimise the static energy consumption, see *lines 5-9*. In other words, balancing the load among the servers (of homogeneous resources) may not add a considerable advantage to energy minimisation as having more active servers for load-balancing means to consume extra static energy.

4.4 The Evaluation of the Proposed Schedulers

This section presents the experiments[1] conducted to evaluate the proposed best-effort scheduling approach, concentrating on measuring its effects on the potential energy saving. In particular, we focus on examining the following five points:

- Making a broad comparison with an existing scheduling algorithm CMMS [84] that has been implemented in our simulation MCST for evaluation purposes. Similar to our best-effort scheduling, CMMS applies to multi-cloud systems but its objective is different in that it always attempts to find a cloud that offers the minimal earliest finish time. To make a fair comparison with our scheduling, we have only modified CMMS to take into consideration the deadline constraint.
- Measuring the overall rate of energy that one could potentially reduce by applying the proposed schedulers on different workloads of applications,

[1]The details of our experiments, including all the results, have been made publicly available `http://www.cs.le.ac.uk/people/ayya1/MCST/index.html`

compared with the upper bound of the topmost energy consumption when all clouds run at the highest performance (i.e., under the highest CPU frequencies). We use the general objective function cost (cf. Section 4.1.4) to give a fair comparison that includes application rejection and/or violation cases.

- Measuring, more optimistically, the potential energy reduction when the penalty of our suggested objective function is 0 (i.e., in the case where no application rejection and/or violation cases occur during their scheduling/executions). This approximately gives us the topmost rate of energy that can be reduced using the energy saving techniques considered in this thesis. We also investigate the influence of such energy reduction when assuming different percentages for extending application deadlines.
- The effectiveness of the proposed strategies, PRS and CRS, for different workloads of applications with respect to (i) the average reduction of energy and (ii) the number of HPC application rejections and violations.
- The effect of the proposed strategies, PRS and CRS, on the utilisation of cloud resources over time.

4.4.1 Configurations and Input Applications

In order to evaluate our scheduling algorithms, we have built a configurable prototype tool (MCST) that simulates a decentralised multi-cloud system (cf. Chapter 3). The simulation experiments use five distributed clouds around the world. The characteristics of these clouds including approximate distances between them are shown in Table 4.2 and Table 4.3. The energy parameters in Table 4.2 are obtained from the previous work by Garg et al. [53]. In each cloud site, we assume the capacity of VMs per server is twice the number of its physical processors, and all the processors support 5 levels of frequency in $[f_{min}, f_{max}]$,

Data center	**Total #VMs**	**Performance (TFLOP/s)**	**CPU parameters**		
			α	β	f_{max}
WestUSA	32000	0.0072	7.5	65	1.8
Germany	32000	0.0096	60	60	2.4
Japan	32000	0.012	4.5	90	3.0
India	32000	0.0128	4.0	90	3.2
Brazil	32000	0.0128	4.4	105	3.2

TABLE 4.2: Specification of the five clouds used in our simulation.

—	**W**estUSA	**G**ermany	**J**apan	**I**ndia	**B**razil
W	—	9094.4	8632.4	13365.2	9058.6
G	9094.4	—	9058.5	6759.7	9442.2
J	8632.4	9058.5	—	5965.9	17389.8
I	13365.2	6759.7	5965.9	—	16201.9
B	9058.6	9442.2	17389.8	16201.9	—

TABLE 4.3: Approximate distances in km between the cloud data-centres.

Category	**Max.** n_{t_i}	**# applications**	**# tasks in each** *app*
Low-load	8696	200	64
Mid-load	11384	200	64
High-load	16384	200	64

TABLE 4.4: Three categories of parallel workload applications.

where f_{min} is 37.5% of f_{max} (we adopt this percentage from [53]). For the speed of the network signal between these five clouds, we assume that the data travel at a rate of $200,000$ km/s.

Table 4.4 describes three categories of parallel application workloads extracted from different logs of real large-scale systems (i.e., LLNL-uBGL-2006-0, LLNL-Thunder-2006-0, LLNL-Atlas-2006-0, and ANL-Intrepid-2009-1) [95, 51]. The task dependencies are inferred, as in [84], from the provided start and finish times of executing jobs in each log. All tasks of app_m are CPU-bound. The applications are submitted to the multi-cloud system at different times, and the gap interval between each two consecutive application submissions is equal to 1000 seconds.

To evaluate the EGSBE and ELS algorithms with both PRS and CRS strategies, we consider the highest frequency mode as an upper bound of energy usage. It models the case when the objective of the cloud providers is to offer their services at the highest performance. It is also applied with PRS (referred to as PRS.HF mode) and CRS (referred to as CRS.HF mode) to attempt minimising energy usage. We assume that Rt is equal to 0.01 for the PRS strategy to allow choosing the cloud that gives the minimum energy immediately without much consideration of $occupRt$.

In addition to the general objective function as a metric (cf. Section 4.1.4), the rate of total energy usage is calculated to compare the different scheduling strategies by $\frac{\sum_{c_j \in C}(E_{c_j}+T_{c_j})}{\sum_{app_m \in \mathcal{A}'} \sum_{t \in app_m}(n_t \cdot v_t)}$. Here, E_{c_j} and T_{c_j} are the amount of energy usage by cloud c_j for execution and transmission activities, $\mathcal{A}'$ is the set of applications that are successfully completed, and $(n_t \cdot v_t)$ is the computing volume of the executed task t that belongs to a successfully completed application app_m.

4.4.2 Effect of the Proposed Schedulers on Energy Optimisation

In Figure 4.5, Figure 4.6, and Table 4.5, we report the main results obtained by running our simulation (MCST) 15 times independently. Each run is performed based on a specific scheduling strategy (i.e., either PRS, CRS, PRS.HF or CRS.HF) in addition to CMMS, and uses 200 HPC-applications, each consisting of 64 tasks, as inputs from one of the workload categories (low-load, mid-load or high-load), described in Table 4.4. The assumed deadline to execute these applications is calculated by their estimated execution time $eExecTime_{app_m}$, extended by 20%, i.e., $DL_m = eExecTime_{app_m} + (eExecTime_{app_m} \times 0.2)$, where the DL_m is the application deadline.

Unsurprisingly, increasing workload categories from low to high would increase linearly the required amount of energy. However, in Figure 4.5, Figure 4.6,

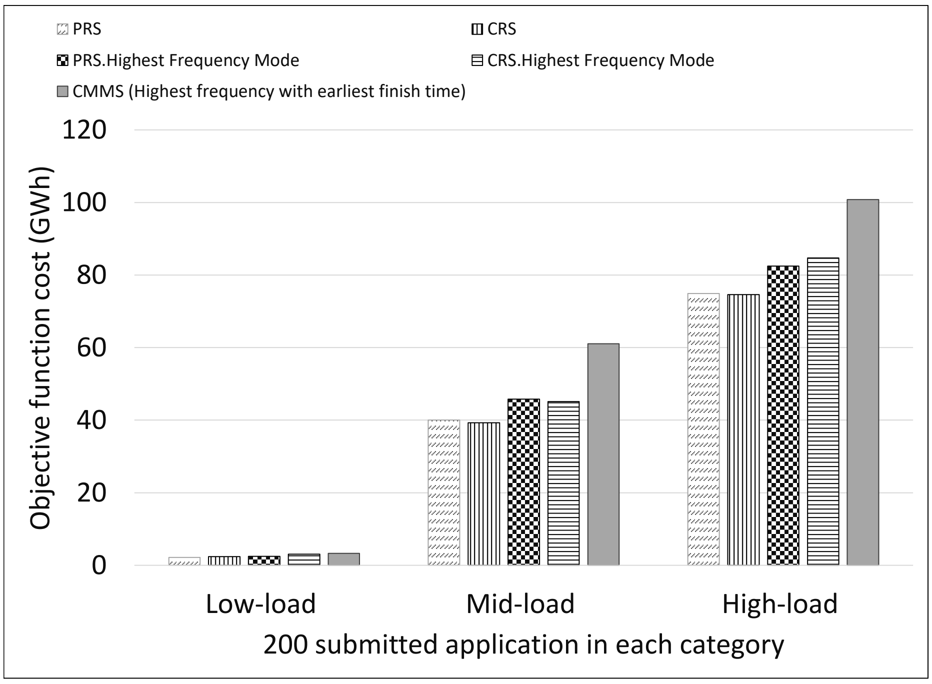


FIGURE 4.5: Different scheduling strategies with various workload categories vs. the objective function.

we pay specific attention to the comparison between PRS/CRS and CMMS, and also between PRS and PRS.HF as well as between CRS and CRS.HF to figure out how much the proposed solutions could reduce energy consumption. Additionally, these figures show differences between the two suggested scheduling strategies themselves, PRS and CRS, explaining which of them gives better results depending on the mentioned workload categories. Comprehensively, Figure 4.5 illustrates the overall cost achieved by PRS, CRS, PRS.HF, CRS.HF, and CMMS using the objective function (cf. Section 4.1.4) that considers the penalty costs for application rejection/violation cases. Figure 4.6, however, gives more details from the same conducted experiments but it concentrates on the rate of energy usage. Apart from CMMS, Figure 4.7 gives other details regarding the unsuccessful cases of executing applications that have occurred when applying PRS, CRS, PRS.HF, and CRS.HF. The values obtained from

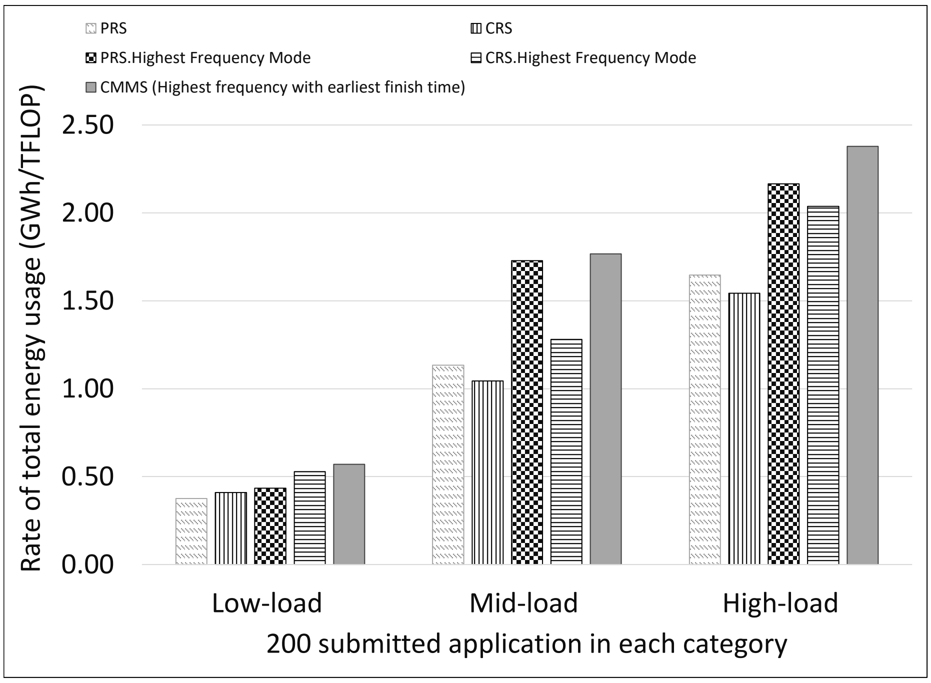


FIGURE 4.6: Different scheduling strategies with various workload categories vs. the rate of energy usage.

these experiments are broken down in Table 4.5.

Observing the different energy costs, shown in Figure 4.5, the overall impact of the proposed schedulers on energy optimisation is positively clear. In comparison with the CMMS algorithm, our scheduling (CRS) allows one to efficiently reduce the total amount of energy in all cases of workloads by an average of approximately 29% without sacrificing desired performance, expressed by meeting application deadlines. Furthermore, both strategies, PRS and CRS, allow to consume less energy than PRS.HF and CRS.HF, in all loads of application categories, by an average of about 11.8% and 15.8%, respectively. Moreover, there appears to be a clear difference between PRS and CRS in the total energy cost, such that PRS outperforms CRS in the low-load case only by roughly 8.3%. However, CRS performs slightly better than PRS in both mid and high-loads by about 1.9% and 0.3%, respectively.

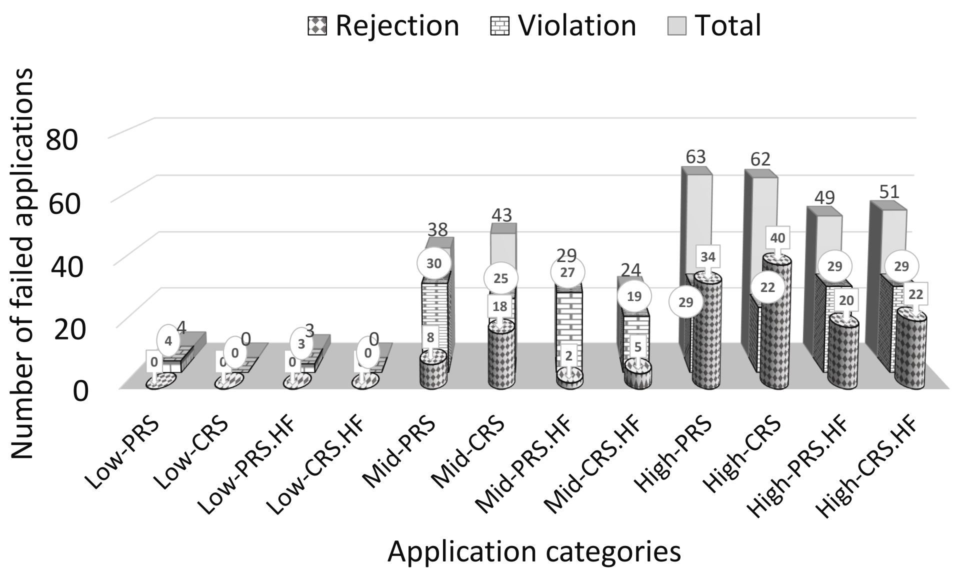


FIGURE 4.7: Different scheduling strategies with various workload categories vs. the number of failed applications.

Figure 4.6 also illustrates different energy rates obtained by PRS, CRS, PRS.HF, CRS.HF, and CMMS for executing application tasks as well as transmitting their datasets between clouds. The significant point here is that while CRS did consume higher energy rate than PRS in low-load case, it efficiently executes all the submitted applications with no failed cases, compared to 4 violated applications by PRS, see Figure 4.7. Additionally, CRS performs better than PRS, in the mid and high-load applications, as it has consumed a lower energy rate by about 7.9% and 6.2% respectively. This points out that using the size of the workloads as a factor to determine the difference between PRS and CRS is not accurate, meaning that analysing more deeper factors regarding the specification of the submitted applications is significant.

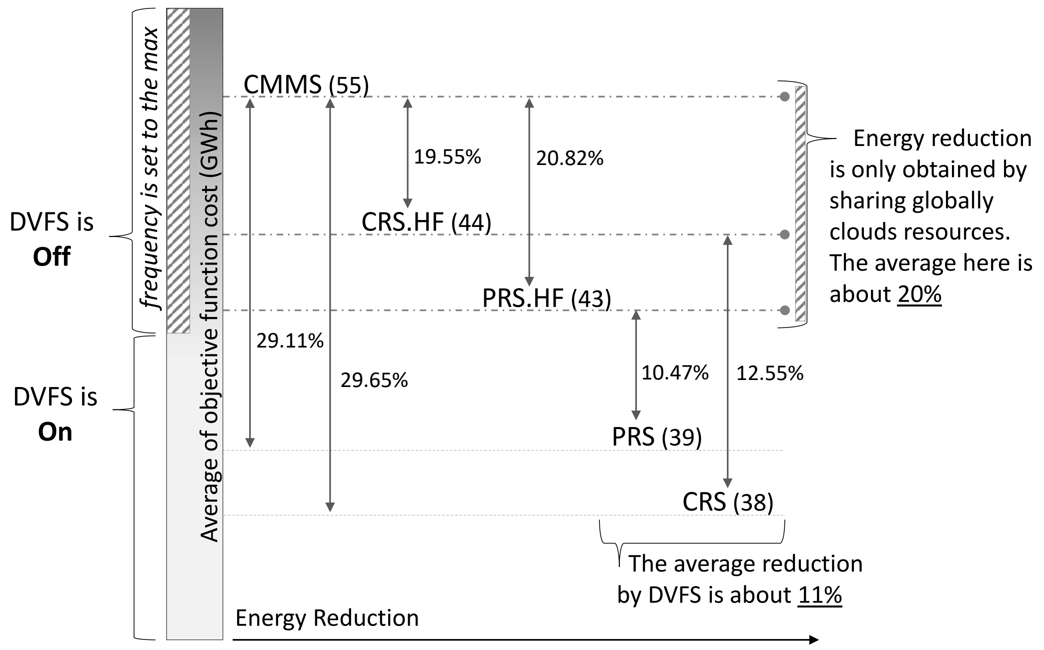
Having a deeper look into the results, shown in Figure 4.5 and Table 4.5, we could estimate the amount of energy that has been reduced by each of the considered energy saving methods, i.e., DVFS as well as sharing globally available

		Computing	Energy cost in GWh			Unexe-
	Strategies	**vol. TFLOP**	**Computing**	**Transm.**	**Penalty**	**cuted**
Low-Load	PRS	5756605	2155186	3718	2045	4
	PRS.HF	5743162	2492222	1624	3510	3
	CRS	5759295	2351578	4955	0	0
	CRS.HF	5759295	3037271	3925	0	0
	CMMS	5759295	3279559	2326	0	0
Mid-Load	PRS	9686959	10980076	2062	29066356	38
	PRS.HF	12409219	21449248	3340	24360068	29
	CRS	15903788	16599582	4976	22649335	43
	CRS.HF	24064282	30806849	5959	14342824	24
	CMMS	27044606	47764229	18374	13293320	64
High-Load	PRS	10509191	17298988	1095	57590138	63
	PRS.HF	14624576	31660135	8199	50819699	49
	CRS	11676748	18017073	7326	56568605	62
	CRS.HF	17365642	35375942	10976	49299236	51
	CMMS	24695519	58743425	11346	42069656	85

TABLE 4.5: Details of the energy usage and other values obtained from scheduling three different workloads vs. different strategies, based on the best-effort mode.

clouds resources. Since all PRS, CRS, PRS.HF, and CRS.HF rely on our global scheduler EGSBE for reducing energy consumption (i.e., by exploiting the available resources from other heterogeneous clouds), and only PRS and CRS apply DVFS while the CPU frequencies in both PRS.HF and CRS.HF are set at the maximum, we could measure how much the DVFS has positively affected the overall reduction by taking the average of the comparison between PRS with PRS.HF, and also CRS with CRS.HF. In Figure 4.8, we illustrate these differences, pointing out that an average reduction of about 11% is caused by DVFS. Moreover, comparing PRS.HF and CRS.HF with CMMS indicates the possible reduction of about 20% that could be obtained by only the global scheduler as these three scheduling strategies do not enable DVFS. As mentioned earlier in this section, the total reduction obtained by applying the two saving methods,

	Average of objective function cost (GWh)	CRS	PRS.H	CRS.H	CMMS
PRS	39	0.77%	10.47%	11.88%	29.11%
CRS	38		11.16%	12.55%	29.65%
PRS.H	43			1.57%	20.82%
CRS.H	44				19.55%
CMMS	55				



frequency is set to the max
DVFS is Off
DVFS is On
Average of objective function cost (GWh)
CMMS (55)
19.55%
20.82%
CRS.HF (44)
PRS.HF (43)
29.11%
29.65%
10.47%
12.55%
PRS (39)
CRS (38)
Energy reduction is only obtained by sharing globally clouds resources. The average here is about 20%
The average reduction by DVFS is about 11%
Energy Reduction

FIGURE 4.8: An estimation, based on the results presented in Figure 4.6, to break down the overall reduction of the consumed energy into the two considered energy saving methods: DVFS and sharing globally the available cloud's resources. The table above also summarises the comparison between the scheduling strategies in terms of the energy reduction according to the objective function.

as compared to CMMS, is around 29%. However, from this conducted experiment (see, Table 4.4), we cannot conclude, in absolute terms, that the idea of utilising globally the available resources from other clouds would always outperform the DVFS in saving energy or even vice versa. These energy saving techniques are always affected by many variables, mainly applications deadlines as well as resource availability. In any case, the focus in our approach is to take the total reduction by both saving techniques regardless of which one may outperform the other.

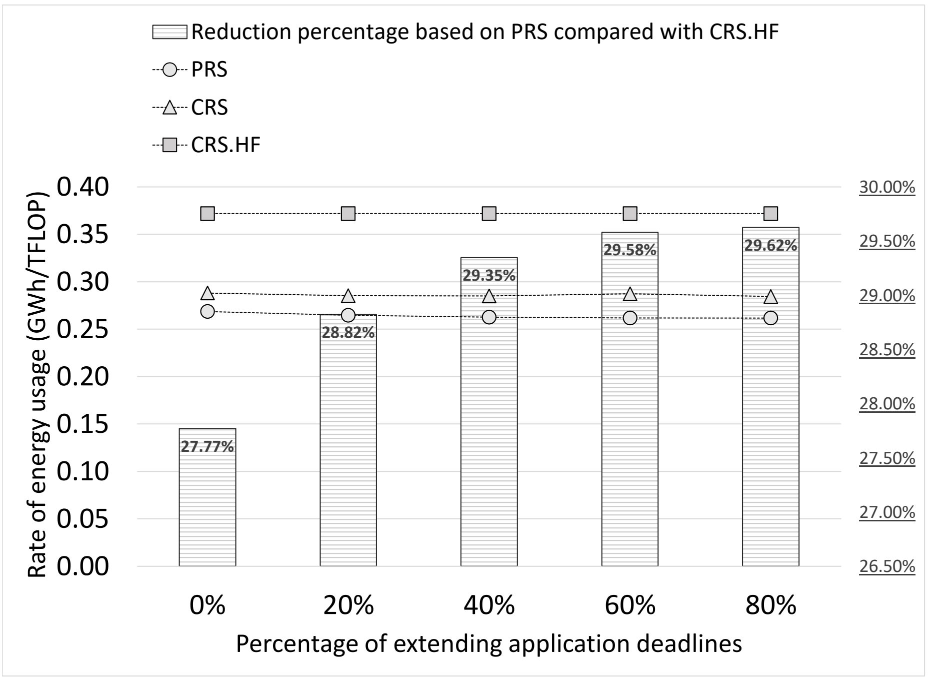


FIGURE 4.9: Different scheduling strategies with 50 submitted applications from the mid-load category vs. the extension of application deadlines in increments of 20%.

Another experiment was conducted to determine the possible energy reduction that could be obtained when the penalty of the considered objective function (cf. Section 4.1.4) is 0, i.e., in the case of having no rejection/violation cases during the schedule and/or the execution of the submitted applications. Although this experiment gives more optimistic estimation, it shows a little more accurate result as it does not include any amount of energy wasted by the partial execution of violated applications. In other words, consider a submitted application where some of its tasks have been successfully executed but, later on, a successor of one of the successfully executed tasks has failed to execute. Such an application is considered *violated*, and the penalty for it (i.e., based on our objective function) is calculated at the level of the application, which includes the cost for both the successfully executed as well as the unexecuted tasks.

To get zero penalty cost, the experiment is performed on 50 submitted applications instead of 200, resulting in no rejection/violation cases occurring during running our simulation (MCST). These applications are randomly chosen from the mid-load category (see Table 4.4), and then successfully executed by MCST 15 times based on PRS, CRS and the highest performance mode (CRS.HF) as shown in Figure 4.9. For each scheduling strategy, we extend the deadline to execute these applications in increments of 20%, starting from the tight deadline (i.e., no extension is applied). From Figure 4.9, we observe that the use of the proposed scheduling strategies can offer reducing energy up to 27% in the case of tight deadlines. In addition, further energy reduction seems to be achievable by extending application deadlines up to 40%, but no significant reduction is expected from 60% extension onward.

4.4.3 Impact of the Proposed Scheduling Strategies on Resource Utilisation

Figure 4.10 and Figure 4.11 depict the utilisation of all clouds in the system when allocating tasks that belong to 200 high-load submitted applications by running PRS and CRS, respectively. Here, we can observe the behaviours of resource occupation in the multi-cloud system. PRS always exploits heavily the cloud that gives the lowest energy (e.g., see *India* and *WestUSA* for the tasks 1 - 937), causing in some cases the applications scheduled later to be allocated to less efficient clouds. As CRS tries to balance the level of resource utilisation over all clouds, it chose the most efficient clouds in a balanced way since the beginning and utilised the worst one (i.e., *Brazil*) when it was forced to do so because of the lack of resources, see Figure 4.11. Note that the global scheduler over heterogeneous clouds shows a better energy performance with load-balancing strategy (CRS) at some level of high resource utilisation. This

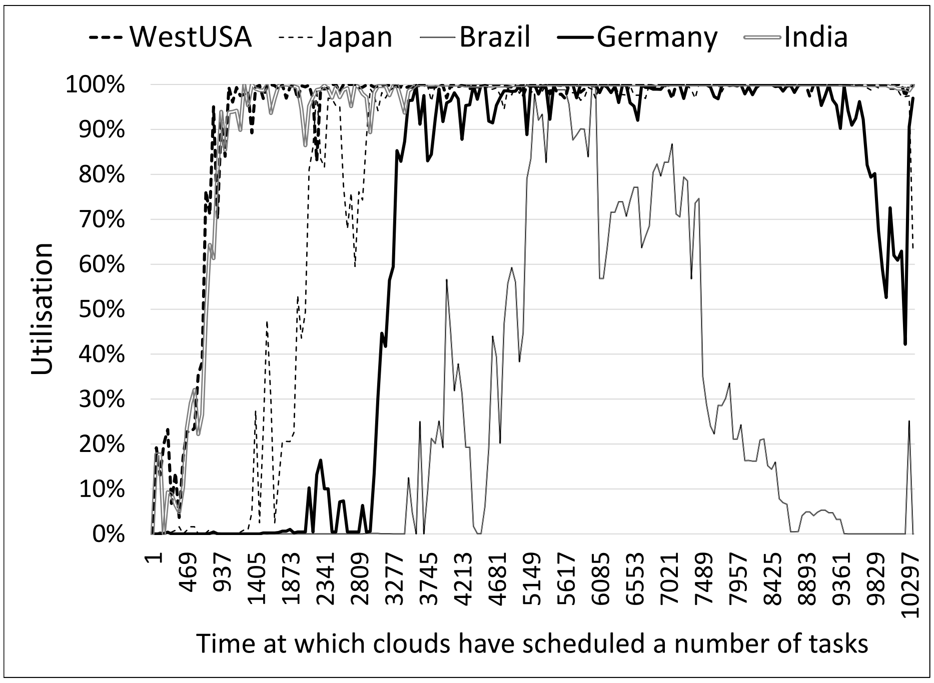


FIGURE 4.10: Resource utilisation with preference rate Strategy PRS.

is, in fact, opposite to what the local scheduler ELS attempts to do where its goal is not to balance the load but to occupy the active servers to the maximum.

4.4.4 Discussion and Experiment Findings

In the following three points, we briefly discuss the main findings observed by our experiments, presented in this chapter.

- There appears to be a considerable amount of energy wasted by clouds that focus only on providing high performance services for executing HPC-applications as suggested by CMMS. Without affecting the desired performance, represented by meeting the specified deadlines to execute applications, such a problem of wasting energy can be, with a great chance, addressed. Based on our observation, using DVFS as well as sharing

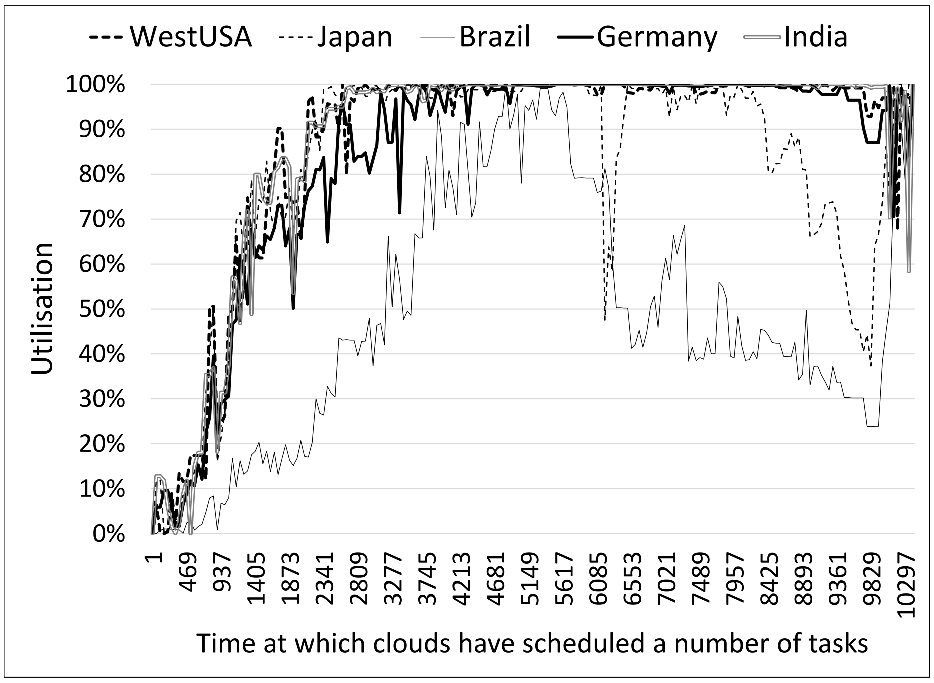


FIGURE 4.11: Resource utilisation with combination rate strategy CRS.

globally cloud resources as techniques for saving energy while scheduling HPC-applications based on best-effort style allows us to reduce a significant amount of energy that could reach approximately 29% compared with the consumption of the high performance. According to the conducted experiments in this chapter, this percentage can be roughly broken down into 10% and 19% for DVFS and sharing globally cloud resources respectively. However, the focus in this thesis is to consider the total reduction by both saving methods, while scheduling HPC-applications, regardless of which one may give better reduction than the other.

- Considering the overall reduction of energy, CRS gives better results than PRS for medium to high load of 200 applications. Whereas PRS seems generally better with low load applications, and also with a certain number of medium load applications (i.e., less than 200) that particularly do not mostly occupy all cloud resources. From this observation, we derive that

– regardless of the size of the workloads – if the overall level of resource occupations over all participating clouds is below average, balancing cloud resources by CRS would not be of such importance, and hence, applying PRS may give more efficient results. On the contrary, CRS probably fits in well with the high levels of resource occupations as balancing comprehensively all cloud resources helps to keep clouds from running out of resources. Accordingly, an intelligent decider for applying dynamically the proper strategy (i.e., either PRS or CRS) based on the kind of workload, cloud resources, and application deadlines is important. This dynamic decider might lead to achieving further energy optimisation.

- Assume a negotiational approach for submitting HPC-applications between cloud's users and cloud's providers to be settled before accepting user's submissions. For example, cloud's users can specify a deadline constraint for executing their applications through the cloud interface, and thereafter, cloud providers can suggest a set of extensions to be made with some extra discount offers as a next option to choose from, allowing cloud's users to rethink about their specified deadlines. For applying such a negotiational approach when scheduling HPC-applications based on the proposed best effort mode, the deadline extension to execute the applications as a pre-defined constraint is suggested to be set up to 40% as there appears to be a very minor benefit whenever extending the deadline longer than 50%.

4.5 Summary

The energy consumption of multi-cloud systems that focus on providing high performance services for executing HPC-applications can be efficiently reduced without affecting the desired performance. Hence, in this chapter, we have introduced and evaluated a proposed energy aware scheduling approach that follows the best-effort scheduling mode. The approach takes advantage of the possibility of sharing multi-cloud resources for executing dependent HPC tasks as well as the adoption of dynamic voltage and frequency scaling (DVFS), as two levels of energy saving techniques. The suggested best-effort scheduling approach analyses: (1) the available resources from all clouds, and (2) the energy costs for executing as well as transmitting HPC tasks between clouds. It comprises two strategies for scheduling decisions: *Preference Rate* which needs to be pre-defined, and *Combination Rate* that works dynamically, aiming to address conflicting objectives using a statistical approach.

We have conducted several experiments using various parallel application workloads extracted from different logs of real large-scale systems. The results have confirmed the possibility of saving some wasted energy of approximately 29.9% compared to clouds that are not concerned about energy consumption (e.g., clouds that adopt the CMMS algorithm) but concentrate on offering high performance services only. We have also given a detailed comparison between the proposed strategies, PRS and CRS, depending on three different workloads of applications as well as the utilisation of cloud resources over time. Both strategies produce mixed results, and it was difficult to determine the absolute best in all cases, including the size of workload, occupied level of cloud resources, and constraint deadline of applications. Nevertheless, achieving further energy optimisation is still possible if a dynamic decider that could determine the proper strategy (i.e., either PRS or CRS) at a specific state of the multi-cloud system is applied.

As an alternative approach to the best-effort mode for scheduling HPC-applications between clouds, we investigate in the next chapter the effect of the global scheduling part using advance-reservation mode on the overall energy optimisation accompanied with the minimisation of application rejection/violation cases.

Chapter 5

Energy Optimisation with Advance-Reservation Mode

In an analogous approach to scheduling HPC-applications using the best-effort mode, presented in Chapter 4, this chapter presents a comparable scheduling technique but based on the advance-reservation mode. In principle, dynamic scheduling with the advance-reservation mode, using the token technique, depends on the actual available slots for a given period of future time. This scheduling technique needs to ensure that the time frame of an application is scheduled within the resource capacity, taking into account already occupied and reserved resources. Apart from the case of unexpected resource failures, this scheduling technique is not expected to violate deadlines when executing an already scheduled application due to time or resource conflicts.

This chapter introduces the second original contribution to optimising energy consumption when scheduling HPC-applications over a distributed multi-cloud system, published in [5]. It proposes *EGSAR*, a token-based reservation algorithm as a novel energy-aware scheduling for allocating a submitted application to the best cloud in the system. *EGSAR* supports minimising application violation cases based on the proposed CRS strategy (cf. Section 4.2). It considers gathering two energy costs when globally scheduling an application, which are:

(i) execution energy at the CPU level and (ii) dataset transmission energy (if applicable) at the network level. The proposed advance-reservation scheduling approach depends on *ELS* (presented in Section 4.3.2) for scheduling application tasks locally in cloud resources (i.e., mapping tasks to machines) using the dynamic voltage and frequency scaling to reduce energy consumption.

The chapter will refer frequently to four shared parts with Chapter 4 as follows:

- The energy model, in Section 4.1, for calculating the energy consumption by executing and transmitting HPC-applications. It includes a formal explanation of our main aims, represented by an objective function, see Section 4.1.4.
- The suggested preference and combination rate strategies (PRS and CRS), in Section 4.2, for choosing the best available resources, that are offered by a set of clouds. PRS allows us to optimise energy consumption based on setting an upper limit for the allowed energy consumption, and CRS attempts to address conflicting objectives using a statistical approach.
- The local scheduler *ELS* algorithm proposed for mapping application tasks to CPUs, introduced in Section 4.3.2.
- Experiment setup and configurations that are defined in Section 4.4. They primarily include (1) the declaration of a number of cloud instances as well as the characteristics of their resources. (2) Different loads of HPC-applications instances that were generated from real parallel workload applications.

However, the new parts are organised into four sections as follows. An illustrative scheduling scenario and problem formulation are given in Section 5.1. The proposed algorithms are explained in Section 5.2, and then evaluated in

Section 5.3. In Section 5.4, we give a detailed comparison between the best-effort and the advance-reservation scheduling approaches before summarising the chapter.

5.1 Definition and Illustration of Scheduling Problem

As mentioned previously in Section 3.1, the system model considered in this thesis focuses on a decentralised multi-cloud system, consisting of geographically distributed heterogeneous clouds that participate in a federated approach. Following this model, we present our scheduling framework based on the advance-reservation mode, which includes the global and the local schedulers. Unlike our best-effort approach for distributing and scheduling tasks to clouds, introduced in Chapter 4, the global scheduler here works at application level, i.e., distributing and assigning the whole application to a cloud. Specifically, as we consider advance-reservation scheduling of dependent tasks, it would not be efficient to schedule them separately to different clouds. This is due to the fact that the advance-reservation is always reliable in executing applications within the reserved time window, and no violation is expected if no failure occurs in machine/resources. Hence, a poor reliability in meeting application deadlines is highly expected when executing globally each (dependent) tasks as sending/receiving datasets (outputs of predecessors) via the global communication is not always guaranteed to be on-time. Thus, the more reliable method is to schedule each whole application to one cloud. The local scheduler for mapping application tasks to CPUs, however, is the same as in the best-effort approach, see Section 4.3.2.

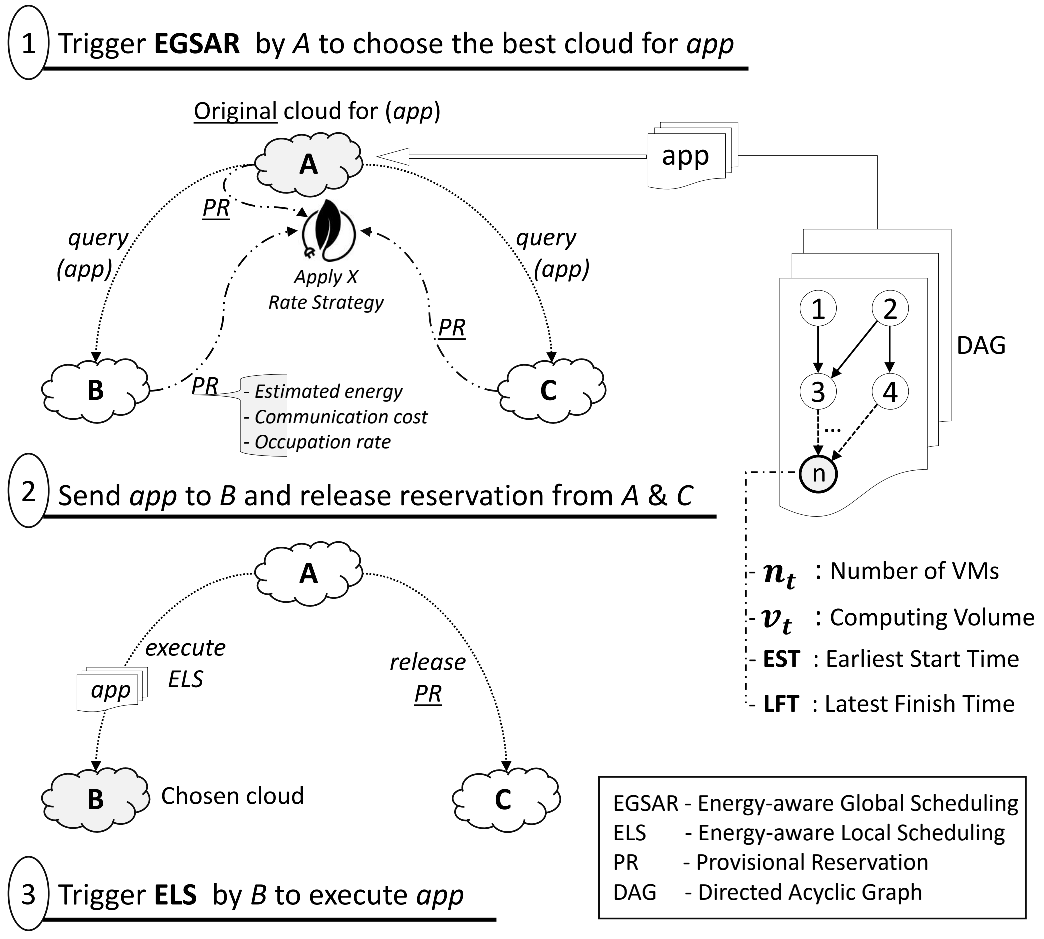

1 Trigger EGSAR by A to choose the best cloud for app
Original cloud for (app)
A
app
query (app)
PR
Apply X Rate Strategy
B
C
- Estimated energy
- Communication cost
- Occupation rate
DAG
1
2
3
4
n
- n_t : Number of VMs
- v_t : Computing Volume
- EST : Earliest Start Time
- LFT : Latest Finish Time
2 Send app to B and release reservation from A & C
execute ELS
release PR
Chosen cloud
3 Trigger ELS by B to execute app
EGSAR - Energy-aware Global Scheduling
ELS - Energy-aware Local Scheduling
PR - Provisional Reservation
DAG - Directed Acyclic Graph

FIGURE 5.1: Overview of the proposed advance-reservation approach, described in three steps.

Figure 5.1 gives an overview of our token-based scheduling, illustrating the role of *global* and *local* schedulers when they receive a submitted application. Consider the cloud A that receives an application *app* with its specific requirements for execution, i.e., A becomes the 'original cloud' for *app*. The *global scheduler* (manager server) in A broadcasts query-messages to all participating clouds (i.e., to B and C) as well as checking the local resources via its *resource controller*. In response to such queries, the local *resource controller* of each cloud provides a provisional reservation PR that consists of (1) estimated execution energy, (2) estimated energy for data transmission and (3) resource occupation rate, see B in Figure 5.1. These PRs are then analysed by the original cloud

A, using the EGSAR algorithm, for deciding which cloud provides the best option for executing *app*. Assume B is the chosen cloud (see the bottom left of Figure 5.1). In this case, A will send messages to release *PRs* from all unchosen clouds, and concurrently it sends the whole application *app* to the chosen cloud B. Here, the *local scheduler* of B applies the scheduling algorithm ELS (discussed in Section 4.3.2) for mapping tasks to machines, taking into account *app*'s requirements and its precedence constraints.

Scheduling Framework. Our framework for scheduling submitted applications permits energy optimisation, in the first place, without affecting the desired performance. However, it is required that the deadline DL_m can be met by at least one participating cloud. In general, each submitted app_m can be either scheduled and then executed successfully, violated, i.e., scheduled but failing to get enough resources at execution time, or rejected.

The framework relies on the energy model, introduced in Section 4.1, for calculating the cost of executing app_m and transmitting its data. On top of this, it applies the two scheduling strategies (preference-rate and combination-rate) for deciding the cloud that provides the best option, discussed in Section 4.2. Each cloud, including o_{app_m}, that meets DL_m should provide a provisional reservation (PR), consisting of:

- i Estimated processing energy for all t_i.
- ii Estimated data transmission energy for the input, the output and the disk image of app_m.
- iii Resource occupation rate from ST_m to DL_m.

Given a set of PRs, the preference-rate strategy first selects a subset of this set by checking the maximum allowable energy of (i) and (ii) of each PR provided, then it gives priority to (iii). Combination-rate strategy, however, analyses (i), (ii) and (iii) of all provided PRs by dynamically adjusting the priority between estimated energy and occupation rate. Along with energy optimisation, it aims to avoid violation cases that may be caused by unexpected resource failures. An application app_m is rejected if all clouds provide a negative PR. This means all clouds in the system do not have enough resources to schedule app_m due to either their capacity limit or a tight DL_m.

Problem Formulation. The main objective of the proposed advance-reservation scheduling approach is the same as in the best-effort scheduling approach that is explained formally in Section 4.1.4. Both rely on the general objective function cost to give a fair comparison between potential energy consumption and penalty that includes the cost of application rejection and/or violation cases.

5.2 EGSAR: Energy-Aware Global Scheduling with Advance Reservation

As described in Figure 5.1, EGSAR takes as input a submitted application and a set of provisional reservations PRs from participating clouds to pick the best PR provided. It is triggered by o_{app_m} upon the arrival of all the responses from clouds. The positive responses offer the ability of scheduling the submitted application while meeting its deadline. Each PR consists of both $eEnergyValue$ (representing estimated energy cost for executing the submitted application plus the cost of transmitting its dataset) and *occupRt* and has a limited token period of validity starting from the time of response, which enables a cloud provider to release its resources by cancelling its PR if no confirmation is received from o_{app_m} within the allowed time.

Algorithm 3 EGSAR
Energy-aware global scheduling with advance-reservation.

```
Inputs: $app$ is a submitted application to $o_{app}$
        $C = \{c_1, \cdots, c_k\}, k \in \mathbb{N}$ is a set of connected clouds
        $PRS$ (preference) and $CRS$ (combination) are rate strategies.
Outputs: A globally scheduled $app$ in $c_i$, the best cloud found.
Begin
 1: $PRs := \emptyset$ is a set of provisional reservations provided from all $c_i$
 2: for each $ms_i$ cloud manager server $\in C$ do
 3:    $pr \leftarrow ms_i$.makeReservation($app$) expanded in Algorithm 4
 4:    if $pr.eEnergyValue > 0$ and $pr.Token.isValid$ then
 5:       $PRs \leftarrow PRs \cup \{pr\}$
 6: if $PRs == \emptyset$ then
 7:    set $app$.isRejected=true
 8: else if $PRs.size == 1$ then
 9:    set chosenCloud($app, PRs(1)$)
10: else
11:    set $pr \leftarrow$ applyDecisionStrategy ($PRs$, $CRS$ (or $PRS$))
12:    chosenCloud ($app$, $pr$)
13:    removeFrom ($PRs$, $pr$)
14:    releasePR ($app$, $PRs$)
End
```

The pseudo code presented in Algorithm 3 and 4 gives a high-level view of our EGSAR algorithm, which is performed based on one of the two proposed strategies: PRS or CRS (cf. Section 4.2). This algorithm has been implemented in our simulation (MCST), see Section 3.2 for more technical details. A participating cloud that is able to schedule an application will provide a PR, consisting of a positive estimation of $eEnergyValue$ for processing and transmitting the dataset, see *lines 1-5* of Algorithm 3. A negative $eEnergyValue$, however, means that the provider cloud cannot satisfy the application's deadline, and thus the returned pr will not be added in the list PRs. In *lines 6-9*, the algorithm determines the output of either rejecting *app* if none of the clouds can schedule it (i.e., $PRs == \emptyset$), or selecting a cloud immediately if only one positive option is found (i.e., $PRs.size == 1$). If more than one cloud can execute the *app*, the decision, based on either PRS or CRS, of which cloud will execute

Algorithm 4 makeReservation

```
Inputs: $app_m(V_m, E_m, ST_m, DL_m)$.
Outputs: A provisional reservation PR that should have a valid token
Begin
 1: $PR.eEnergyValue := 0$ and $PR.occupRt := 0$ for initialising $PR$.
 2: $CPUcap :=$ is the processors capacity of this cloud.
 3: for $i := 1$ to $q$ do
 4:     $Thr := CPUcap - n_{t_i}$
 5:     update $EST_{t_i}$ according to its scheduled $pred(t_i)$.
 6:     $P :=$ is the number of used processors at $EST_{t_i}$.
 7:     $bestIntvl := \emptyset$
 8:     if $Thr < 0$ then
 9:         $PR.eEnergyValue := 0$, and then return $PR$.
10:     if $Thr \geq P$ then
11:         $inPeak := false$ and $startAv := EST_{t_i}$.
12:     else
13:         $inPeak := true$.
14:     get all overlapping tasks and form list of start and end points $PT = \{pt_1, \cdots, pt_b\}$,
        excluding any start point $pt$ where $pt \leq EST_{t_i}$.
15:     for $j := 1$ to $b$ do
16:         if $pt_j$ is a start point then
17:             $P = P + n_{pt_j}$
18:             if $inPeak = false$ and $(P > Thr$ or $pt_j = LFT_{t_i})$ then
19:                 try to schedule $t_i$ in $[startAv, pt_j]$ and get its $f$
20:                 if $f$ is the minimum frequency in this cloud then
21:                     $bestIntvl := [startAv, pt_j]$ then break
22:                 else if $[startAv, pt_j]$ is longer than $bestIntvl$ then
23:                     $bestIntvl := [startAv, pt_j]$
24:                 $inPeak := true$
25:         else
26:             if $pt_j$ is an end point then
27:                 $P = P - n_{pt_j}$.
28:                 if $inPeak = true$ and $P \leq Thr$ then
29:                     $inPeak := false$ and $startAv := pt_j$.
30:     if $bestIntvl \neq \emptyset$ then
31:         schedule $t_i$ to $bestIntvl$.
32:         calculate the estimated energy consumption $E_{t_i}$.
33:         $PR.eEnergyValue = PR.eEnergyValue + E_{t_i}$
34:     else
35:         $PR.eEnergyValue := 0$, and then return $PR$.
36: if $\neg$ isOriginalToThisCloud($app_m$) then
37:     calculate the estimated energy for transmitting dataset $T_{app_m}$ using (4.4).
38:     $PR.eEnergyValue = PR.eEnergyValue + T_{app_m}$
39: calculate $PR.occupRt$ by dividing the total number of required CPUs for all occupied/re-
    served tasks in cloud resources between the start and finish times of executing $app_m$ by
    $CPUcap$.
40: generate unique $token$ and associate it with $PR$.
41: return $PR$.
End
```

it is described in *lines 11-12* of Algorithm 3. In *lines 13-14*, all unchosen clouds are notified to release their pr.

The aim of Algorithm 4 is to reserve the required number of processors for all tasks in the submitted application app_m at the best possible interval found within the range of the EST and the LFT of each task while minimising the energy usage. These EST and LFT are calculated using (3.1) and (3.2) respectively, such that the start time ST_m (i.e., equal to the submission time) will be assigned to EST_{t_i} of all entry tasks, whereas the specified deadline DL_m with the submitted application will be assigned to EST_{t_i} of all exit tasks. Nevertheless, for each non-entry task t_j, the EST_{t_j} will be adjusted just before t_j is getting scheduled according to the actual finish time of all its $pred(t_j)$, see *line 5* of Algorithm 4.

Considering closely Algorithm 4, if the reservations for all tasks have been made, the algorithm will return a provisional reservation PR that should have a valid token associated with the app_m in *line 41*, it returns a negative PR (expressed by $PR.eEnergyValue = 0$) in *line 9* or *35*, otherwise. The main loop, *line 3*, goes over all tasks of app_m, and for each task t_i, the algorithm attempts to determine all the peaks within the cloud resources between the times (EST_{t_i} and LFT_{t_i}), see *lines 4-14*. Here, each peak represents a top time point pt of resource utilisation, associated with occupied/reserved resources by all the overlapping tasks that belong to different applications, such that after pt the level of resource utilisation will get down. In *lines 15-29*, the algorithm checks the possibility of scheduling t_i between each two consecutive peaks at which t_i can be executed. During checking these peaks, it remembers the previous possible scheduling option, as each time it attempts to find a slower frequency as a better option. If t_i can be scheduled, its estimated energy cost will be calculated to be added to $PR.eEnergyValue$ as shown in *lines 30-33*. If the current cloud is not the original for app_m, the cost of transmitting the dataset of app_m from the original cloud to the current one will be calculated using (4.4), and then adding this cost to $PR.eEnergyValue$ as shown in *lines 36-38*. In *lines 39* and *40* the algorithm calculates the $PR.occupRt$, and then generates a token to be associated with the successful reservation. Finally, the algorithm will return the PR to Algorithm 3 as shown in *line 3*.

5.3 Experimental Evaluation

This section presents the experiments[1] conducted to evaluate the proposed schedulers, concentrating on three aspects:

- Measuring the effect of the proposed schedulers on the energy saving based on the general objective function cost (i.e., the total of energy usage plus penalty for rejected/violated applications). It gives the average reduction, depending on different application workloads, compared with the total energy obtained by running all clouds at the highest performance (i.e., under the highest CPU frequencies)
- Comparing the advance-reservation scheduling with an existing scheduling algorithm CMMS [84] that aims to choose a cloud that offers the minimal earliest finish time regardless their expected energy consumption. We have slightly modified the CMMS to take into consideration the deadlines constraint of submitted applications as well as to make a provisional reservation with an estimated early finish time for each application, in order to make a valid comparison.
- Measuring the impact of resource failures (i.e., failures that may occur accidentally in the cloud system) on already scheduled applications. This is to get a rough idea of how our proposal can contribute to providing a reliable scheduler.

[1]The details of our experiments, including all the results, are available at `http://www.cs.le.ac.uk/people/ayya1/MCST/index.html`

In addition, this section will discuss the effectiveness of the proposed strategies, PRS and CRS that are introduced in Section 4.2, for different workloads of applications with respect to (i) the average reduction of energy and (ii) the number of HPC application rejections and violations.

5.3.1 Configurations

In this experiment, we have used the same configurations that were defined for evaluating the best-effort scheduling approach, presented in Section 4.4. More precisely, the simulation (MCST) experiments use a decentralised multi-cloud system of five worldwide distributed clouds, and the characteristics of these clouds including approximate distances between them are shown in Table 4.2 and Table 4.3. Furthermore, we have made the same assumptions, mainly:

- the capacity of VMs per server is double the number of its physical processors in each cloud site and all the processors allow to change frequency among 5 levels ranging between $[f_{min}, f_{max}]$, where f_{min} is 37.5% of f_{max} (we adopt this percentage from [53]);
- the HPC-applications are submitted to the original cloud at different times, and the gap interval between each two consecutive application submissions is equal to 1000 seconds. In addition, all tasks of these applications are CPU-bound; and
- the formula $(\frac{\sum_{c_j \in C}(E_{c_j}+T_{c_j})}{\sum_{app_m \in \mathcal{A}'} \sum_{t \in app_m}(n_t \cdot v_t)})$ is defined to calculate the rate of the total energy usage by all clouds in the system, allowing to compare the results obtained by the two proposed scheduling strategies. The E_{c_j} and T_{c_j} are the amount of energy usage by cloud c_j for execution and transmission activities, $\mathcal{A}'$ is the set of applications that are successfully completed, and $(n_t \cdot v_t)$ is the computing volume of the executed task t that belongs to a successfully completed application app_m.

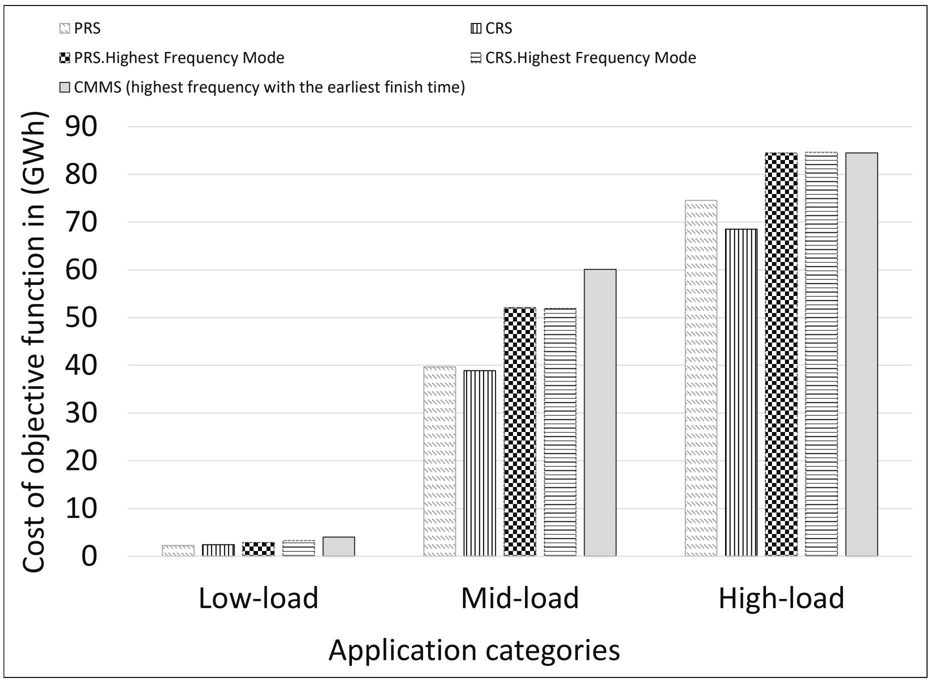


FIGURE 5.2: Different scheduling strategies with various workload categories vs. the objective function.

For defining HPC-applications as input to MCST, we have also used the logs of parallel application workloads shown in Table 4.4. We assume the deadline to execute a submitted app_m is calculated by its estimated execution time $eExecTime_{app_m}$, extended by 20% in the case of loose deadline, and by 0.1% for tight deadline. For example, the loose deadline DL_m is calculated as $DL_m = SubmissionTime_{app_m} + eExecTime_{app_m} + (eExecTime_{app_m} \times 0.2)$, where the $SubmissionTime_{app_m}$ is assumed to be equal to the start time ST_m.

Similar to our way of evaluating the EGSBE and ELS algorithms, the highest frequency mode is considered as an upper bound of energy usage to assess how much the EGSAR algorithm can optimise energy consumption. We apply the highest frequency mode with both strategies, PRS (referred to as PRS.HF mode) and CRS (referred to as CRS.HF mode). For the PRS strategy, *Rt* is also assumed to be 0.01, which allows to pick the cloud that offers the lowest energy immediately without taking *occupRt* into consideration.

	Strategies	Computing vol. TFLOP	Energy cost in GWh Computing	Transm.	Penalty	Unexe-cuted
Low-Load	PRS	5759295	2204252	33321	0	0
	PRS.HF	5759295	2811659	67137	0	0
	CRS	5759295	2296750	123360	0	0
	CRS.HF	5759295	3037514	199609	0	0
	CMMS	5759295	3287565	721576	0	0
Mid-Load	PRS	14788565	16842446	117473	22686841	23
	PRS.HF	22924935	35265693	190969	16629230	16
	CRS	15940101	16052168	181244	22634057	21
	CRS.HF	23343210	34999483	180379	16737415	16
	CMMS	22123895	41307065	698839	18103817	16
High-Load	PRS	14550372	21007754	101307	53460288	44
	PRS.HF	20483726	36198549	90839	48199162	38
	CRS	15023757	13216840	187977	55089309	43
	CRS.HF	20691843	36881432	167611	47581003	38
	CMMS	20282422	33641959	639189	50198049	39

Table 5.1: Details of the energy usage and other values obtained from scheduling three different workloads vs. different strategies, based on the advance-reservation mode.

5.3.2 Effect of the Proposed Schedulers on Energy Optimisation

Figure 5.2 and Table 5.1 show the total energy cost achieved by PRS, CRS, PRS.HF, CRS.HF, and CMMS according to the proposed objective function that is applied on the various workloads with loose deadlines. The majority of the submitted applications have been successfully executed, and all of the rest have been rejected, i.e., no violated application cases occurred. For all cases of application workloads, our scheduling, relying on both PRS and CRS, enables to efficiently optimise energy consumption as compared with CMMS that focuses on selecting clouds that only offer earliest finish time for executing applications without looking at energy cost. The average energy reduction that could be

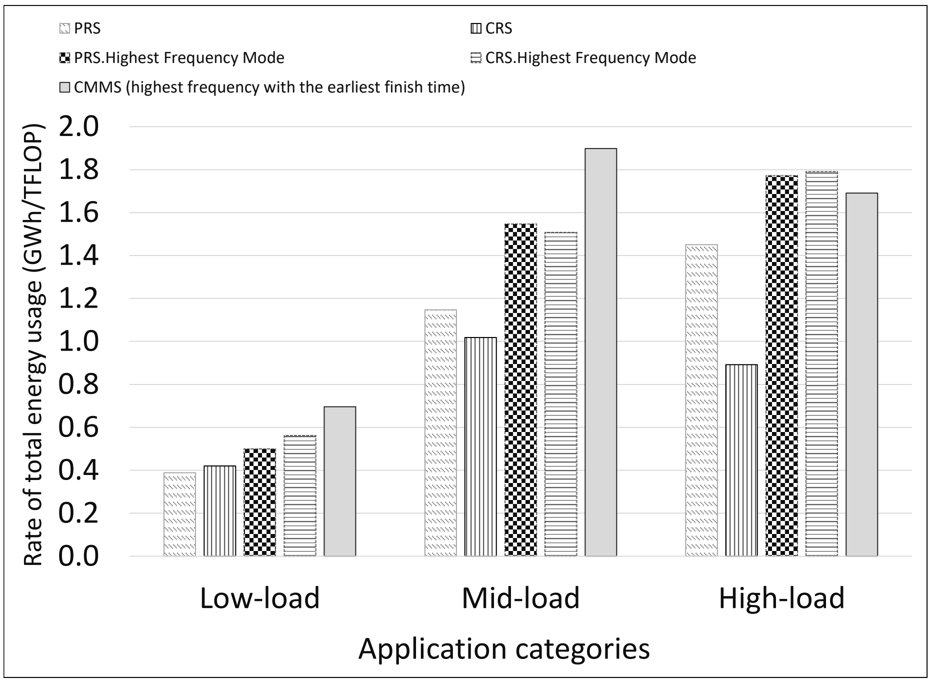


FIGURE 5.3: Different scheduling strategies with various workload categories vs. the rate of energy usage. The deadlines of executing all these applications have been extended based on loose deadlines.

gained without sacrificing desired performance is roughly 31.3%, calculated by comparing the results obtained from CRS vs. CMMS.

On the one hand, it is clear from Figure 5.2 that PRS and CRS produce a lower energy cost than PRS.HF and CRS.HF in all cases by an average of about 19.3% and 23.1%, respectively. On the other hand, the chart shows a considerable difference in the total energy cost of PRS and CRS, with PRS being smaller than CRS by about 7.5% in the low-load case. However, CRS produces total energy cost less than the one by PRS by about 1.97% with mid-load and about 8.15% with high-load. This indicates that whenever the workload gets heavier the CRS strategy produces better results in terms of total energy cost.

Considering the rate of the energy consumed by executing tasks and transmitting datasets depending on the loose deadline, shown in Figure 5.3, the difference between PRS, CRS, PRS.HF, CRS.HF, and CMMS is still evident with the

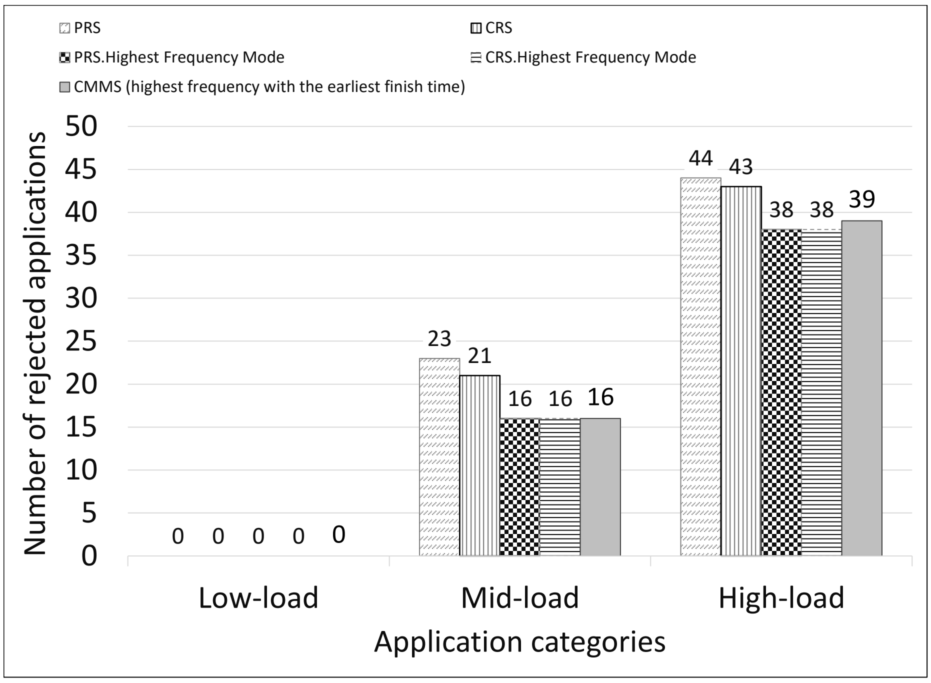


FIGURE 5.4: Different scheduling strategies with various workload categories vs. the number of rejected applications. The deadlines of executing all these applications have been extended based on loose deadlines.

various workload categories. Despite the fact that CRS computed more volume than PRS due to a lower number of rejected applications with both mid-load and high-load, as shown in Figure 5.4, it has a lower rate of energy usage than PRS by an average of about 24.8%. Moreover, PRS.HF and CRS.HF produced different rates of energy usage, although they rejected the same number of applications with the different workload categories. The values obtained from these experiments are broken down in Table 5.1.

To estimate the potential energy reduction that one could obtain in the case of no rejection/violation cases occurring during scheduling or executing submitted applications, we have performed an experiment similar to the one presented in Figure 4.9 but based on the advance-reservation mode. The idea is to estimate a little more precise cost of energy consumption by having 0 penalty in the objective function (cf. Section 4.1.4). This experiment was conducted on 50

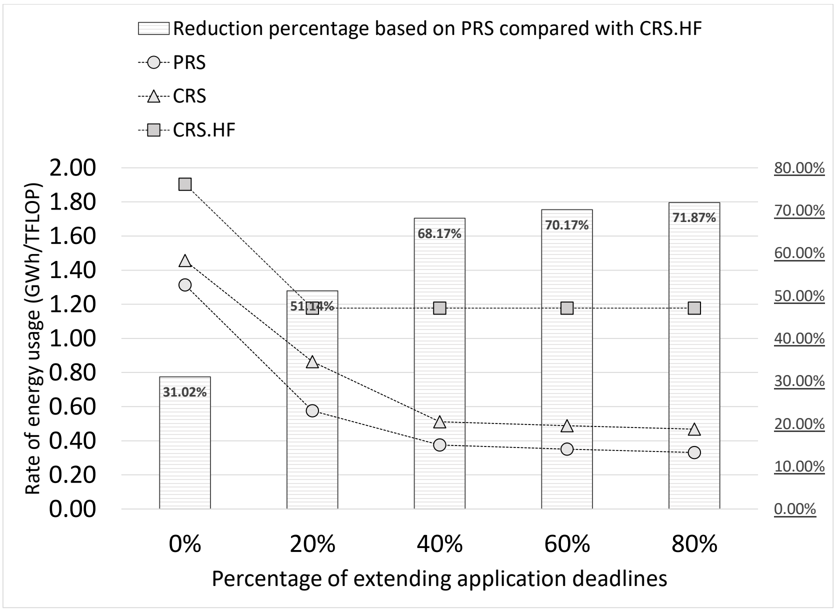


FIGURE 5.5: Different scheduling strategies with 50 submitted applications from the mid-load category vs. the extension of application deadlines in increments of 20%.

submitted applications that were randomly chosen from the mid-load category (Table 4.4). All these applications have been successfully executed 15 times, by our simulation (MCST), based on the strategies (PRS, CRS and the highest performance mode CRS.HF), such that for each scheduling strategy, we extend the deadline in increments of 20%, starting from 0% to 80%. The result is shown in Figure 5.5, which indicates the significant effect on reducing energy consumption by PRS and CRS if the deadline of the submitted applications is allowed to be extended up to approximately 40%.

5.3.3 Impact of the Resource Failures on Scheduled Applications

To evaluate the robustness of our scheduling, with respect to resource reliability, against unforeseen technical failures on cloud servers, we simulate our

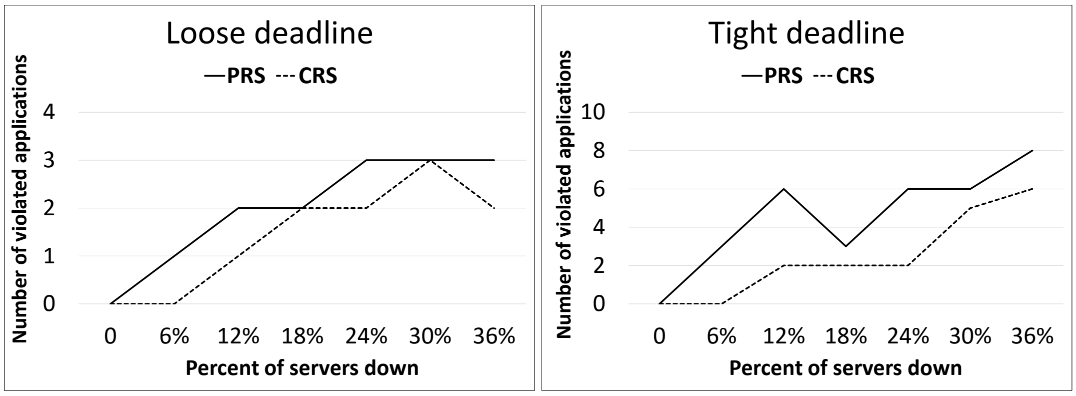


FIGURE 5.6: Number of violations vs. percentage of down servers.

advance-reservation approach in scenarios where a percentage of servers (growing in increments of 6%) become unavailable at runtime. For each execution, the failures are triggered from time 0, and stay in a failed state until the end of the simulation. Figure 5.6 shows the number of violations occurring due to the increased number of unavailable servers in all cloud sites for inputs with loose and tight deadlines. The experiments are performed on 40 submitted applications that are randomly mixed from the low-load and mid-load categories. In the case of loose deadlines CRS achieved a number of violations lower than PRS by an average of 36.1%, while it was about 51.4% lower in the tight deadlines case. This reflects the positive effect of the dynamic consideration of our robustness metric CRS as compared to PRS.

5.3.4 Discussion and Experiment Findings

The experimental results illustrate that scheduling HPC-applications, focusing on optimising overall energy consumption, is affected by several interdependent factors. The affecting elements we want to shed light on are the kinds of applications in terms of their requirements and the status of resource occupation when an application is received for scheduling. In general, scheduling an application to a cloud that appears better at the submission time may not lead to the best energy saving result over time. Specifically, in the case of high-load applications, CRS produces better energy savings than PRS due to the dynamic technique of balancing the workloads among all clouds when applicable.

In Figure 5.7 and Figure 5.8, we depict the utilisation of cloud resources when scheduling 200 applications from the high-load category (Table 4.4) based on PRS and CRS, respectively. As observed from the performance of the best-effort scheduling, explained in Section 4.4.3, the PRS often starts picking clouds that offer minimum energy without balancing the level of resource utilisation over all clouds as the CRS always tries. These behaviors are generally identical with the advance-reservation scheduling mode, consider closely *Japan* for the submissions (1 - 28) in Figure 5.7 vs. *Germany* in Figure 5.8. Here, *Japan* seems to consume lower energy than *Germany* as the latter has a higher α, recall the characteristics of these clouds in Table 4.2. Thus, PRS has first exploited *Japan* until the performance of this cloud has reached just above 80%, whereas, the maximum utilisation by CRS over all clouds has reached a percentage that is lower by 10%. The main benefit of balancing the utilisation of cloud resources by CRS is the possibility of avoiding the forced cases of choosing inefficient clouds, i.e., in the case of not having enough resources by efficient clouds. However, if the level of resource utilisation over all clouds is not high, i.e., at or below 50%, applying PRS instead of CRS is certainly suggested. In the following list, we summarise our main findings of scheduling HPC-applications using the advance-reservation mode.

- The best energy cost saving of about 23%, based on our objective function, is obtained by CRS compared to its upper bound CRS.HF.
- None of the strategies (i.e., PRS or CRS) proves itself to be the best with any submitted workloads in terms of energy efficiency, where PRS shows better results with low-load compared to CRS.

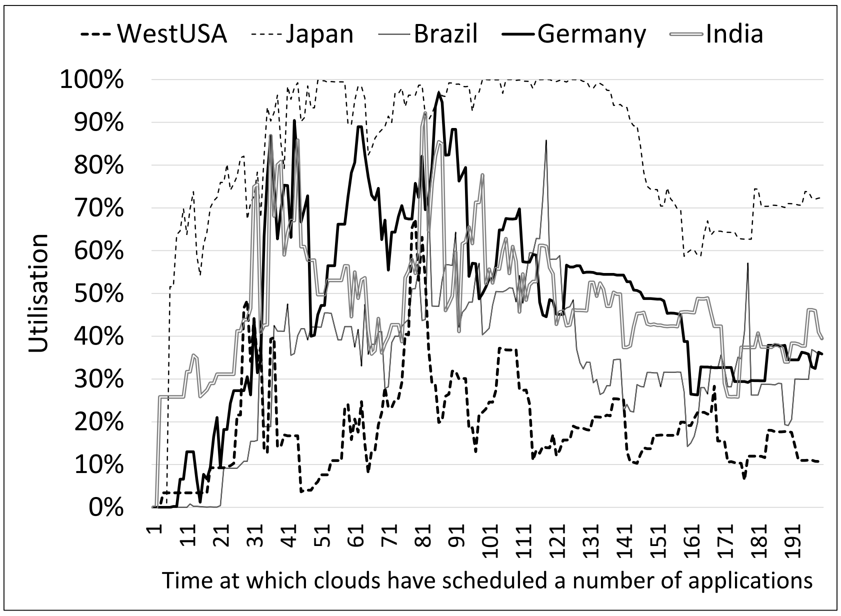


FIGURE 5.7: Resource utilisation with preference rate Strategy PRS.

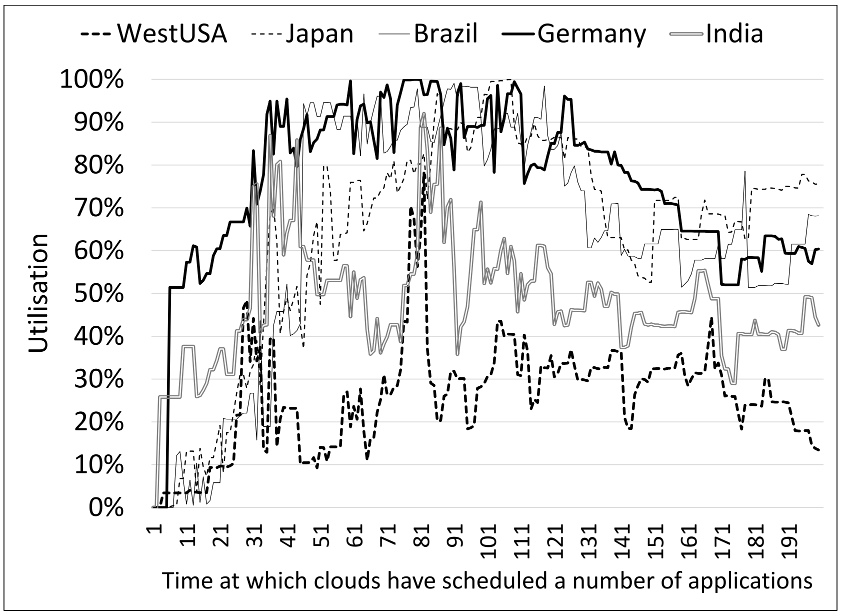


FIGURE 5.8: Resource utilisation with combination rate strategy CRS.

- Deadline violation cases of applications can be reduced for both tight and loose deadlines with CRS being better than PRS by an average of 43%.

- As we use token-based reservation, the looseness of deadlines of the submitted applications is a crucial factor impacting on the number of application rejections.

5.4 Comparison with the Best-Effort Approach

Since the practical experiments conducted to evaluate both the best-effort (cf. Section 4.4) and the advance-reservation (cf. Section 5.3) approaches have used the same configuration as well as the same application inputs (cf. Section 2.2.2) by our simulation (MCST), it should be valid to compare their overall outputs for more critical evaluations. As the local scheduler of both approaches is the same, the comparison here will concentrate on the global scheduling parts only that aim to find the best participating cloud to execute a task (by the best-effort) or an application (by the advance-reservation). In particular, we concisely discuss which of the proposed global scheduling approaches generally outperforms the other in terms of the following three points:

- The overall energy optimisation as well as the scheduling efficiency represented by the avoidance of application rejections and violations.
- The utilisation of the network between clouds, expressed by the number of messages that are generated by MCST as SimJava events (cf. Section 3.2).
- The effect of extending application deadline on the rate of energy reduction.

Besides, we have made the comparison relying on the proposed strategies (PRS and CRS, introduced in Section 4.2) and the three workload application categories (low, mid and high, presented previously in Table 4.4).

In order for the comparison of energy consumption to be fair between the proposed scheduling approaches, the overall energy rates are considered, which are computed after finishing the run of the simulation (MCST). Each energy rate is the total amount of energy (for executing application tasks as well as transmitting their datasets between clouds) divided by the total computing volumes of all applications that have been successfully executed. More specifically, we compare the energy rates obtained by two separate experiments, reported on Figure 4.6 and Figure 5.3, as these experiments have been conducted based on the same given 200 HPC-applications. As an illustration, Figure 5.9 describes the energy rate of the comparison between the global scheduling approaches, based on the strategies PRS and CRS. It clearly shows that both scheduling modes, the best-effort and the advance-reservation, give slight differences in the low and mid workloads. In the high workload, however, the advance-reservation outperforms the best-effort in both strategies, PRS and CRS, by approximately 20% and 65% respectively. As a result, Figure 5.9 reveals that applying the scheduling based on the CRS and the advance-reservation mode is generally better for energy optimisation, as they would enable to either produce a better or slightly different outcomes than could be obtained by the best-effort mode and/or the other strategy PRS.

In addition, the advance-reservation scheduling mode provides more efficient results in terms of minimising application rejections/violations, see the number of unsuccessfully executed applications out of the 200 submissions in Table 5.2. While reserving and then releasing cloud resources for specific token times often has a consequence on accurately determining the best available clouds, this technique seems better than the attempt of using unguaranteed cloud resources (i.e., without making prior reservations) for minimising application rejection and/or violation cases. Additional insight concerning the potential avoidance of rejecting applications or failing to execute scheduled tasks by applying the CRS strategy can be deduced from the results in Figure 4.7 and Figure 5.4.

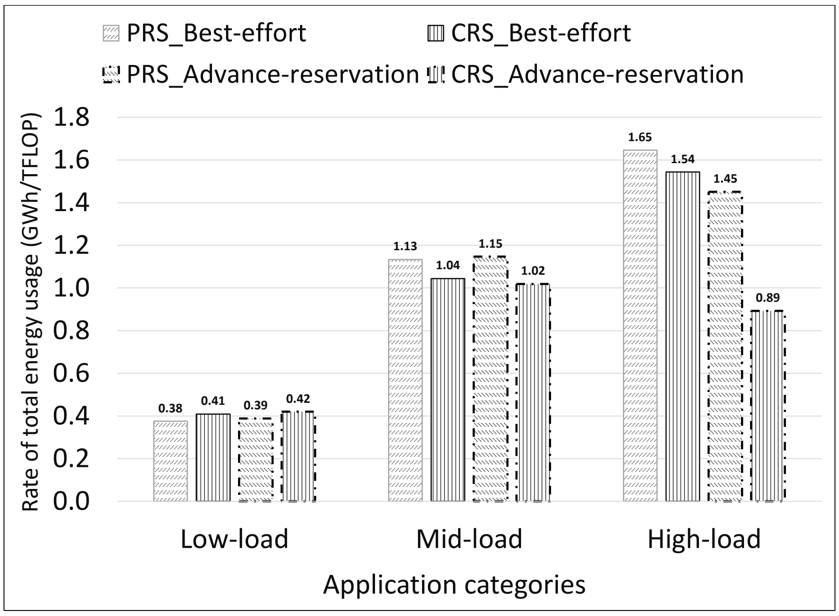


FIGURE 5.9: Comparison between the best-effort and the advance-reservation scheduling modes in the rate of energy usage based on the preference and combination rate strategies.

	Rejection	Violation	Total
Low-PRS.BE	0	4	4
Low-CRS.BE	0	0	0
Low-PRS.AR	0	0	0
Low-CRS.AR	0	0	0
Mid-PRS.BE	8	30	38
Mid-CRS.BE	18	25	43
Mid-PRS.AR	23	0	23
Mid-CRS.AR	21	0	21
High-PRS.BE	34	29	63
High-CRS.BE	40	22	62
High-PRS.AR	44	0	44
High-CRS.AR	43	0	43

PRS	Preference Rate Strategy.
CRS	Combination Rate Strategy.
BE	Best-effort scheduling mode.
AR	Advance-reservation scheduling mode.

TABLE 5.2: Comparison between the best-effort and the advance-reservation scheduling modes in the number of rejected and violated applications based on the preference and combination rate strategies.

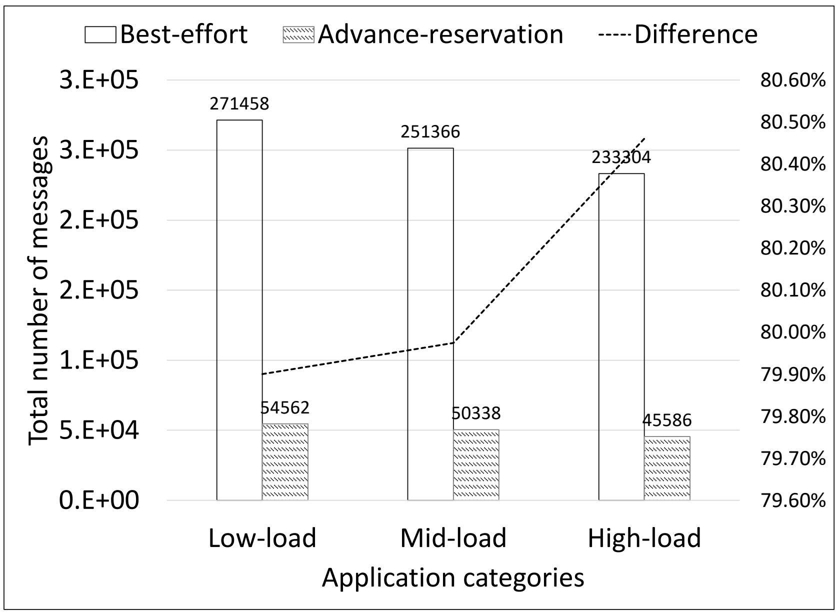


FIGURE 5.10: Comparison between the best-effort and the advance-reservation scheduling modes in the network utilisation, represented by the number of messages that are generated by MCST as SimJava events.

To help understand the difference between the best-effort and the advance-reservation scheduling modes in the utilisation of the global network (i.e., between participating clouds), we have identified the total number of messages that are sent between clouds for a complete run of our simulation (MCST), including query messages and messages for sending datasets, using *SimJava.Sim_event*. Clearly, the advance-reservation would utilise the global network much less than the best-effort as the latter attempts to find the best cloud for executing each task, i.e., not the whole application as in the advance-reservation. The results shown in Figure 5.10 estimate that the difference could approximately reach 80%. This could suggest that applying the advance-reservation over an inefficient global network (e.g., one that has a weak performance or only supports some limited speeds) is much better than using the best-effort.

Comparing the effect of extending application deadlines on reducing energy consumption, between the proposed scheduling modes, we concisely show the overall differences in Figure 5.11, obtained from Figure 4.9 and Figure 5.5.

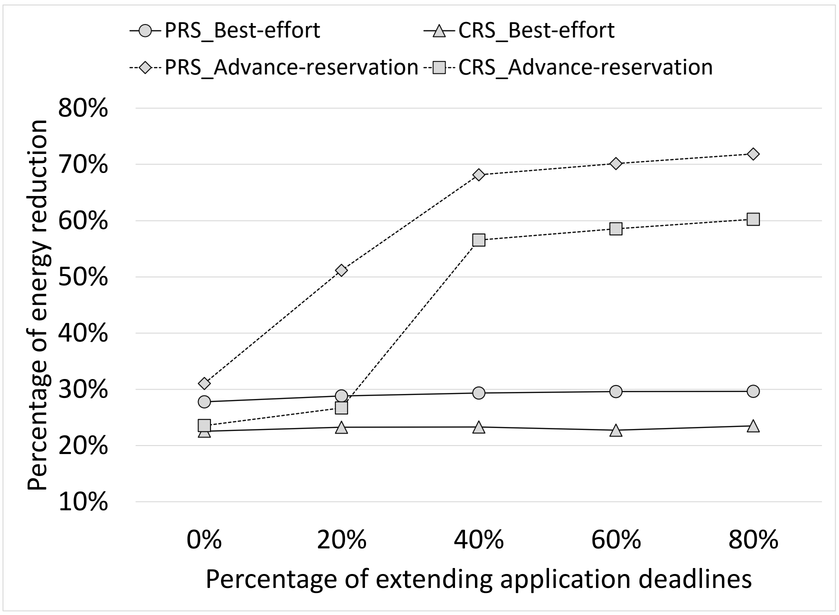


FIGURE 5.11: Comparison between the best-effort and the advance-reservation scheduling modes in the percentage of energy reduction based on the preference and combination rate strategies.

Broadly, the advance-reservation allows to reduce roughly 70% of energy when extending application deadlines to 60%, compared to just 30% by the best-effort, in the case of applying PRS strategy. Additionally, Figure 5.11 points out that extending application deadlines while following the best-effort scheduling mode often tends to be less effective compared with the advance-reservation mode that offers a considerable energy reduction.

5.5 Summary

The aim of this chapter was first to present a reservation based scheduling approach for executing HPC-applications by a multi cloud system, and then compare it with the previous proposed scheduling approach (i.e., the best-effort approach introduced in Chapter 4). We have explained the shared parts between the two approaches, particularly, the local scheduler ELS that aims to allocate

HPC-tasks to CPUs. The main original parts of this chapter, however, have focused on presenting and evaluating the global scheduler over distributed clouds (EGSAR) that relies on the token reservation based technique.

The conducted experiments using our simulation (MCST) have shown that the reservation based scheduling approach using the suggested strategies can reduce energy consumption by (1) an average of 21% as compared to the upper bound, determined by the highest performance possible in the cloud-system, and (2) a better average of 31.3% as compared to the existing algorithm CMMS [84]. The results have revealed that a significant issue in energy aware scheduling is that a designed mechanism that depends only on the absolute minimum energy value to execute an application may not necessarily produce the best overall energy saving in all cases. In general, a considerable amount of energy can be saved with HPC-applications that have flexible deadlines, nearly a reduction of 70%, if the applications allow extension to be applied up to 60%. Furthermore, the results have shown that there is an interdependency between using one of the proposed strategies for scheduling decisions and the characteristics of the submitted applications.

Importantly, the overall impression gained from the comparison between the best-effort and the advance-reservation scheduling approaches has indicated that the latter offers better results in terms of:

- reducing energy consumption; and
- offering more reliable scheduling, expressed by the avoidance level of application rejections and violations.

Thus far, the focus of the suggested approaches was on the technical aspects. Therefore, in the next chapter, we will shift our focus to the theoretical aspects. Specifically, we will analytically study a simplified version of the energy-aware scheduling problem, the so-called speed-scaling scheduling problem, based on the competitive analysis approach (cf. Section 2.3.4).

Chapter 6

Non-Preemptive Scheduling on Power-Heterogeneous Processors

As mentioned in Section 2.3.4, one of our concerns in this thesis is to apply the theoretical analysis to the proposed energy-aware online scheduling methods (EGSBE and EGSAR), which allows us to deeply understand the results from different angles. Generally, here we investigate a simplified form of the scheduling problem under consideration using a different study approach. The scheduling problem considered in the previous two chapters, for allocating HPC-applications while minimising the energy consumption subject to deadline and precedence constraints, is very difficult to be analysed due to its complications. For example, in cloud computing, jobs with various requirements need to be dispatched to servers in a data-centre, where different servers may have different power functions, or in a federated cloud system with different data-centres. We thus, in this chapter, consider a simplified form of the problem that captures some of the main aspects to draw an analogy with our energy-aware scheduling problem in multi-cloud systems that are:

- heterogeneous parallel processors, where the exponent α of the power function can be different for different processors;

- non-preemptive jobs (tasks) that have release time, work, and deadline;
- online scheduling style; and
- the objective is to find a feasible schedule that minimises the total energy consumption by all processors.

It is worth mentioning that precedence constraints are not considered, nor arbitrary release times and deadlines, in this simplified form of the problem as they will make the analysis even more complex and challenging. Consequently, we aim to analyse a non-preemptive online scheduling problem for allocating independent tasks with deadline constraints to heterogeneous processors. This problem fundamentally represents a non-preemptive version of the classical speed-scaling scheduling problem that was studied by Yao et al. [118] but on parallel processors.

This chapter presents the first online algorithms for the non-preemptive scheduling of jobs with agreeable deadlines on heterogeneous parallel processors, published in [7]. The following section shows the problem definition. In Section 6.2, we observe that a variation of AVR can be used to schedule jobs with agreeable deadlines non-preemptively on a single processor. In Section 6.3, we first show that the non-preemptive speed scaling problem for heterogeneous processors can be solved optimally by a simple greedy algorithm if all jobs are identical (i.e., have the same release time, deadline, and work). From this we obtain a $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$-competitive algorithm for jobs with agreeable deadlines whose interval lengths and densities differ by a factor of at most 2. For jobs with equal interval lengths and equal densities, the competitive ratio improves to $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$. In Section 6.4, we extend the result to arbitrary jobs with agreeable deadlines and obtain a competitive ratio of $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)\lceil\log D\rceil^{\alpha_m+1}\lceil\log T\rceil^{\alpha_m+1}$, where D is the density ratio and T is the interval-length ratio.

6.1 Problem Definition

We study non-preemptive online scheduling of jobs with release times and deadlines on heterogeneous processors with speed scaling. There are m processors $P_1, \ldots, P_m$. The power function of processor P_i, $1 \leq i \leq m$, is $f_i(s) = s^{\alpha_i}$ for some constant $\alpha_i > 1$. Without loss of generality, we assume $\alpha_1 \leq \cdots \leq \alpha_m$. There are n jobs $J_1, \ldots, J_n$. Each job J_j has a release time r_j, a deadline d_j, and work (size) w_j. The time period from r_j to d_j is called the *interval* of job J_j, and $d_j - r_j$ is called the *interval length*. The *density* of job J_j is $\delta_j = \frac{w_j}{d_j - r_j}$. Jobs arrive online at their release times. Jobs with the same release time arrive in arbitrary order. Each job must be scheduled non-preemptively on one of the m processors between its release time and deadline. The speed of each processor can be changed at any time, and a processor running at speed s performs s units of work per unit time. Our objective is to find a feasible schedule that minimises the total energy consumption of all m processors. The total energy consumption $E(P_i)$ of processor P_i is the integral, over the duration of the schedule, of the power function of its speed, i.e., $E(P_i) = \int_0^H f_i(s_i(t))dt$, where $s_i(t)$ is the speed of processor P_i at time t and H denotes the time when the schedule ends, i.e., when all jobs are completed. The objective value is the total energy cost, $\sum_{i=1}^{m} E(P_i)$. We refer to the scheduling problem with this objective as *minimum energy scheduling*.

We assume that the jobs have *agreeable* deadlines, i.e., a job with later release time also has a later deadline. Formally, if job J_i arrives before job J_j, then $d_i \leq d_j$ must hold. This assumption is realistic in many scenarios and helps to schedule the jobs assigned to a processor non-preemptively. Let the *density ratio* $D = \frac{\max \delta_j}{\min \delta_j}$ be the ratio between maximum and minimum job density, and let the *interval-length ratio* $T = \frac{\max d_j - r_j}{\min d_j - r_j}$ be the ratio between maximum and minimum interval length.

6.2 Non-Preemptive AVR

Our algorithms decide for each job on which processor it should be run and then each processor uses some local scheduling method to determine a schedule for the jobs allocated to it. For the latter problem we adapt the AVR algorithm that was proposed for online preemptive scheduling by Yao et al. [118]. AVR works as follows. We call a job J_j *active* at time t if $r_j \leq t \leq d_j$. At any time t, AVR sets the speed of the processor to the sum of the densities of the active jobs.

Conceptually, all active jobs are executed simultaneously, each at a speed equal to its density. On an actual processor, this must be implemented using preemption, i.e., each of the active jobs runs repeatedly for a very short period of time and is then preempted to let the other active jobs execute.

To get a non-preemptive schedule for jobs with agreeable deadlines, we modify AVR as follows to obtain NAVR (non-preemptive AVR): The speed of the processor at any time t is set in the same way as for AVR, i.e., it is equal to the sum of the densities of all active jobs (even if some of these jobs have completed already). However, instead of sharing the processor between all active jobs, the jobs are executed non-preemptively in the order in which they arrive, which is the same as earliest deadline first (EDF) order because we have agreeable deadlines. We remark that the idea of a transformation of AVR schedules into non-preemptive schedules for jobs with agreeable deadlines was already mentioned in [15] in the context of offline approximation algorithms.

An example comparing AVR and NAVR on an instance with 3 jobs is shown in Figure 6.1. Each job is shown as a rectangle whose width is its interval length and whose height is its density. AVR shares the processor at each time among all active jobs. NAVR uses the same speed as AVR at any time, but dedicates the whole processor first to J_1, then to J_2, and finally to J_3.

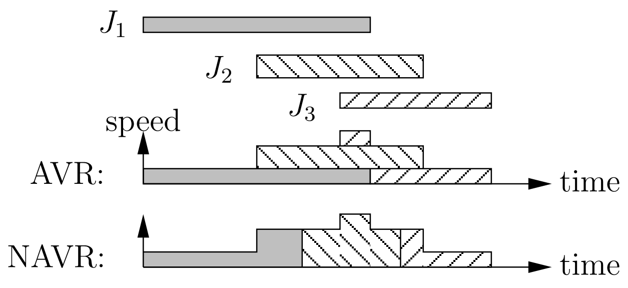


FIGURE 6.1: AVR and NAVR schedules for an example with 3 jobs

Observation 1. For scheduling jobs with agreeable deadlines on a single processor, the schedule produced by NAVR is non-preemptive and feasible. It has the same energy cost as the schedule produced by AVR.

Proof. The schedule produced by NAVR is clearly non-preemptive. It completes all jobs because the total amount of work that is executed is the same as that of AVR as the processor speed is set to the same value in both schedules at all times. For this reason the energy cost is also the same for both schedules. It cannot happen that NAVR starts a job J_j before its release time r_j because AVR executes until time r_j only jobs that arrive before J_j and have deadline before d_j. Therefore, those jobs contain sufficient work to occupy the processor until time r_j. It cannot happen that NAVR finishes a job J_j after its deadline d_j either because all the jobs executed by NAVR until job J_j finishes are jobs that the AVR schedule executes by time d_j. Therefore, the speed of the processor is sufficiently large to complete J_j by its deadline d_j. □

To analyze our algorithms for minimum energy scheduling, we will compare the schedule produced by our algorithms with the optimal schedule that uses AVR (or equivalently NAVR for jobs with agreeable deadlines) on each processor and does not use migration. By reformulating Lemma 8 in [3] for NAVR instead of AVR, we get:

Lemma 6.1. *For instances with agreeable deadlines, there exists a schedule that uses NAVR on each processor and uses energy at most* $(\max_{1\leq i\leq m}\{\rho_i\}+1)OPT$*, where* ρ_i *is the competitive ratio of AVR on processor* P_i*.*

Let OPT^A denote the energy cost of the optimal NAVR schedule (or equivalently the optimal AVR schedule) for a given instance of minimum energy scheduling with agreeable deadlines, and OPT the energy cost of an optimal schedule. By Lemma 6.1 and the result that AVR is $\alpha^\alpha 2^{\alpha-1}$-competitive for a single processor with power function s^α [118], we get the following corollary.

Corollary 6.2. $OPT^A \leq (\alpha_m^{\alpha_m} 2^{\alpha_m-1}+1)OPT$

6.3 Jobs with Small Density Ratio and Interval-Length Ratio

In this section we consider minimum energy speed scaling on heterogeneous processors for job sets where the density ratio and the interval-length ratio are small.

6.3.1 Jobs with Equal Release Time, Deadline, and Density

Let us first consider the special case where all the jobs are identical, i.e., all jobs have the same release time, the same deadline, and the same density. We show that a simple greedy algorithm for allocating the jobs to processors, combined with NAVR on each processor, produces an optimal schedule.

In the proof we will need the following auxiliary result that shows that the extra power required by increasing the speed of a processor by δ grows with the current speed of the processor.

Algorithm 5 Allocation of n identical jobs to m processors

```
for $i \leftarrow 1$ to $m$ do
    $L_i \leftarrow 0$ ;                                   /* load on processor $P_i$ */
end
while not all jobs allocated do
    $J_j \leftarrow$ next job ;                            /* the job has density $\delta$ */
    for $i \leftarrow 1$ to $m$ do
        $Z_i \leftarrow (L_i + \delta)^{\alpha_i} - L_i^{\alpha_i}$;     /* power increase on $P_i$ */
    end
    $i_{\min} \leftarrow \operatorname{argmin}_i Z_i$ ;             /* smallest power increase */
    $L_{i_{\min}} \leftarrow L_{i_{\min}} + \delta$ ;           /* assign job $J_j$ to processor $P_{i_{\min}}$ */
end
```

Lemma 6.3. *Let $\alpha > 1$, let x, y be real values satisfying $0 \leq x \leq y$, and let $\delta > 0$. Then $(x+\delta)^\alpha - x^\alpha \leq (y+\delta)^\alpha - y^\alpha$.*

Proof. Consider the function $g(x) = (x+\delta)^\alpha - x^\alpha$. Its derivative is $g'(x) = \alpha(x+\delta)^{\alpha-1} - \alpha x^{\alpha-1}$, which is positive for $x \geq 0$. This implies that $g(x)$ is monotone increasing for $x \geq 0$, showing that $g(x) \leq g(y)$ for $0 \leq x \leq y$. $\square$

For the given instance with identical jobs, our algorithm assigns the jobs one by one as they arrive, always picking a processor that minimises the increase in power needed to accommodate the extra job. See Algorithm 5 for the pseudo-code.

Lemma 6.4. *Algorithm 5 produces an optimal schedule for identical jobs with respect to energy.*

Proof. Let r be the common release time, d the common deadline, and δ the common density of the jobs. First, observe that if k jobs are assigned to a processor P_i, then the optimal schedule for these k jobs will be to run P_i at speed $k\delta$ from time r to time d and complete the jobs one by one in arbitrary order, with a total energy usage of $(d-r)(k\delta)^{\alpha_i}$ for P_i.

Now, for $1 \leq i \leq m$, let k_i be the number of jobs allocated to P_i by the algorithm, and let o_i be the number of jobs allocated to P_i by the optimal solution. Let ALG denote the total energy cost of Algorithm 5, and OPT the total energy cost of the optimal schedule. We have

$$ALG = (d-r)\sum_{i=1}^{m}(k_i\delta)^{\alpha_i}$$

and

$$OPT = (d-r)\sum_{i=1}^{m}(o_i\delta)^{\alpha_i}\,.$$

Assume that $ALG > OPT$. Then there must be at least one P_i with $k_i > o_i$ and at least one P_h with $k_h < o_h$. Consider the last job, say job J_j, that the algorithm allocated to P_i. At the time the algorithm allocated J_j to P_i, the load of P_h was some $k'_h \leq k_h$. As the algorithm allocated J_j to P_i and not to P_h, we know that

$$(k'_h\delta+\delta)^{\alpha_h} - (k'_h\delta)^{\alpha_h} \geq (k_i\delta)^{\alpha_i} - (k_i\delta-\delta)^{\alpha_i}\,.$$

If we change the optimal schedule by moving one job from P_h to P_i, the energy cost of that schedule increases by $d-r$ multiplied with

$$(o_i\delta+\delta)^{\alpha_i} - (o_i\delta)^{\alpha_i} - \left((o_h\delta)^{\alpha_h} - (o_h\delta-\delta)^{\alpha_h}\right)\,.$$

By Lemma 6.3, we have

$$(o_i\delta+\delta)^{\alpha_i} - (o_i\delta)^{\alpha_i} \leq (k_i\delta)^{\alpha_i} - (k_i\delta-\delta)^{\alpha_i}$$

and

$$(o_h\delta)^{\alpha_h} - (o_h\delta-\delta)^{\alpha_h} \geq (k'_h\delta+\delta)^{\alpha_h} - (k'_h\delta)^{\alpha_h}.$$

This gives:

$$\begin{aligned}(o_i\delta + \delta)^{\alpha_i} - (o_i\delta)^{\alpha_i} - ((o_h\delta)^{\alpha_h} - (o_h\delta - \delta)^{\alpha_h})\\ \le (k_i\delta)^{\alpha_i} - (k_i\delta - \delta)^{\alpha_i} - ((k'_h\delta + \delta)^{\alpha_h} - (k'_h\delta)^{\alpha_h}) \le 0\ .\end{aligned}$$

As we started with the optimal schedule, the change in energy cannot be negative, so the new schedule must have the same energy cost and again be optimal. The total difference between the numbers of jobs allocated to each processor by the optimum and the algorithm, $\sum_i |o_i - k_i|$, has decreased by two in this operation of moving one job in the optimal schedule from P_h to P_i. Hence, this operation can be repeated, without increasing the energy cost, until the optimal schedule and the schedule produced by the algorithm are identical. This proves that the schedule produced by the algorithm is optimal. □

6.3.2 Jobs with Equal Interval Length and Density

We propose an online algorithm for the case where the interval length $d_j - r_j$ is the same for all jobs, and where every job has the same density δ. Note that jobs with equal interval length automatically have agreeable deadlines. To simplify the presentation, we assume w.l.o.g. that $d_j - r_j = 2$ for $1 \le j \le n$.

Algorithm 6 assigns each job to one of the m processors and executes NAVR (cf. Section 6.2) on each processor. Jobs that are released within a time interval $[C, C+1)$ for integer $C \ge 0$ are treated independently from the jobs released in all other such intervals.

Note that Algorithm 6 allocates the jobs arriving in the time period $[C, C+1]$ to processors in the same way as Algorithm 5 would allocate them if they had the same release time. Furthermore, all these jobs are active in the whole interval $[C+1, C+2)$. Lemma 6.4 thus implies the following lemma.

Algorithm 6 Allocation of jobs with equal interval length and density

```
$C \leftarrow 0$ ;                              /* current time period is $[C, C+1)$ */
while not all jobs allocated do
    for $i \leftarrow 1$ to $m$ do
        $L_i \leftarrow 0$
    end
    while next job $J_j$ has $r_j < C+1$ do
        for $i \leftarrow 1$ to $m$ do
            $Z_i \leftarrow (L_i + \delta)^{\alpha_i} - L_i^{\alpha_i}$;          /* power increase on $P_i$ */
        end
        $i_{\min} \leftarrow \text{argmin}_i Z_i$ ;                 /* smallest power increase */
        $L_{i_{\min}} \leftarrow L_{i_{\min}} + \delta$ ;        /* assign job $J_j$ to processor $P_{i_{\min}}$ */
    end
    $C \leftarrow C + 1$
end
```

Lemma 6.5. *Consider the allocation that Algorithm 6 produces for jobs arriving in the time period $[C, C+1)$. Then the energy use for those jobs alone in the time period $[C+1, C+2)$ is less than or equal to the optimal energy cost that any AVR schedule for the same jobs incurs in that period.*

Let ALG_C be the total energy cost of Algorithm 6 in the time interval $[C, C+1)$, and let OPT_C^A be the total energy cost of the optimal AVR schedule in the time interval $[C, C+1)$.

For the schedule of the algorithm, let A_C be the total energy cost incurred during the time period $[C, C+1)$ for jobs that are released in the time interval $[C-1, C)$, assuming that these are the only jobs being executed. Let $k_{C,i}$ be the number of jobs that are released in $[C-1, C)$ and assigned to P_i by the algorithm. Note that we have $A_C = \sum_{i=1}^{m} (k_{C,i}\delta)^{\alpha_i}$. Furthermore, by Lemma 6.4 we have that A_C is the minimum possible energy cost incurred by jobs released in $[C-1, C)$ during the time interval $[C, C+1)$ in any AVR schedule. As the same jobs also execute during $[C, C+1)$ in the optimal AVR schedule, plus possibly some further jobs released in $[C-2, C-1)$ or $[C, C+1)$, we have $A_C \leq OPT_C^A$.

As all jobs have $d_j - r_j = 2$, the jobs that are executed by the algorithm at some point in the time period $[C, C+1)$ are released in one of the intervals $[C-2, C-1)$, $[C-1, C)$ or $[C, C+1)$. Therefore, the total number of jobs allocated to machine i whose execution overlaps $[C, C+1)$ is at most $k_{C-1,i} + k_{C,i} + k_{C+1,i}$. Thus, we have

$$\begin{aligned}
ALG_C &\leq \sum_{i=1}^{m}(k_{C-1,i}\delta + k_{C,i}\delta + k_{C+1,i}\delta)^{\alpha_i} \\
&\leq \sum_{i=1}^{m}(3\delta \max\{k_{C-1,i}, k_{C,i}, k_{C+1,i}\})^{\alpha_i} \\
&\leq 3^{\alpha_m}\sum_{i=1}^{m}\max\{(k_{C-1,i}\delta)^{\alpha_i}, (k_{C,i}\delta)^{\alpha_i}, (k_{C+1,i}\delta)^{\alpha_i}\} \\
&\leq 3^{\alpha_m}\sum_{i=1}^{m}((k_{C-1,i}\delta)^{\alpha_i} + (k_{C,i}\delta)^{\alpha_i} + (k_{C+1,i}\delta)^{\alpha_i}) \\
&= 3^{\alpha_m}(A_{C-1} + A_C + A_{C+1})
\end{aligned}$$

The total energy cost of Algorithm 6 can then be bounded by:

$$\begin{aligned}
ALG &= \sum_{C\geq 0} ALG_C \\
&\leq \sum_{C\geq 0} 3^{\alpha_m}(A_{C-1} + A_C + A_{C+1}) \\
&\leq 3^{\alpha_m} \sum_{C\geq 0} (A_{C-1} + A_C + A_{C+1}) \\
&\leq 3^{\alpha_m+1} \sum_{C\geq 0} A_C \\
&\leq 3^{\alpha_m+1} \sum_{C\geq 0} OPT_C^A \\
&= 3^{\alpha_m+1} OPT^A \\
&\leq 3^{\alpha_m+1}(\alpha_m^{\alpha_m} 2^{\alpha_m-1} + 1) OPT
\end{aligned}$$

where the last inequality holds by Corollary 6.2. Hence, Algorithm 6 is $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$-competitive.

6.3.3 Interval Lengths and Densities Within a Factor of Two

We now consider the case where both the interval length and the density are allowed to vary by a factor of at most two for all the jobs. Here, as we explore the competitive ratio only with the big-O notation analysis, having a different factor e.g. 3, 1.4, etc. rather than 2 would not affect the big-O notation where the competitive ratio will be the same but with a different constant. Thus, choosing the factor of two is a standard choice as long as optimising the constant factor is not considered. In Section 6.4, we will handle the general case by classifying jobs into groups such that each group satisfies this property.

Algorithm 7 Allocation of jobs with interval length in $[y, 2y]$ and density in $[x, 2x]$

```
$C \leftarrow 0$ ;                        /* current time period is $[C, C + \frac{y}{2})$ */
$\delta \leftarrow x$ ;               /* treat all jobs as if their density was $x$ */
while not all jobs allocated do
    for $i \leftarrow 1$ to $m$ do
        $L_i \leftarrow 0$
    end
    while next job $J_j$ has $r_j < C + \frac{y}{2}$ do
        for $i \leftarrow 1$ to $m$ do
            $Z_i \leftarrow (L_i + \delta)^{\alpha_i} - L_i^{\alpha_i}$;          /* power increase on $P_i$ */
        end
        $i_{\min} \leftarrow \text{argmin}_i Z_i$ ;                 /* smallest power increase */
        $L_{i_{\min}} \leftarrow L_{i_{\min}} + \delta$ ;        /* assign job $J_j$ to processor $P_{i_{\min}}$ */
    end
    $C \leftarrow C + \frac{y}{2}$
end
```

Assume that the interval lengths of all jobs are in $[y, 2y]$ and the densities of all jobs in $[x, 2x]$. The algorithm, shown as Algorithm 7, assigns each job to one of the m processors. It treats the jobs as if their density was equal to $\delta = x$ and proceeds in time periods of length $\frac{y}{2}$. Jobs arriving in each time period are handled independently of the jobs arriving in other time periods. On each processor, the allocated jobs are scheduled using NAVR.

Algorithm 7 allocates the jobs arriving in the time period $[C, C+\frac{y}{2})$ to machines in the same way as Algorithm 5 would allocate them if they were identical jobs with density δ. Furthermore, all these jobs are active in the whole interval $[C + \frac{y}{2}, C + y)$ because their interval length is at least y.

Lemma 6.6. *Consider the allocation that Algorithm 7 produces for jobs arriving in the time period $[C, C + \frac{y}{2})$. Then the energy use for those jobs alone in the time period $[C + \frac{y}{2}, C + y)$ is at most 2^{α_m} times the optimal energy cost that any AVR schedule for the same jobs incurs in that period.*

Proof. Let J be the set of jobs arriving in $[C, C + \frac{y}{2})$. Note that their densities are in $[x, 2x]$. Let J' denote the set of jobs obtained from J by setting the density of every job to $\delta = x$. Note that all the jobs in J or J' are active in the whole interval $[C + \frac{y}{2}, C + y)$. Let $ALG(J')$ denote the energy cost of the jobs in J' in the interval $[C + \frac{y}{2}, C + y)$ for the allocation produced by the algorithm, and let $OPT^A(J')$ denote the optimal energy cost of the same jobs in the optimal AVR schedule. Define $ALG(J)$ and $OPT^A(J)$ similarly for J instead of J'. Lemma 6.4 implies that $ALG(J') = OPT^A(J')$. Furthermore, we clearly have $OPT^A(J') \leq OPT^A(J)$. For the algorithm to execute the jobs using their real densities instead of density $\delta = x$ it needs to increase the speed of every processor by at most a factor of 2. Therefore, $ALG(J) \leq 2^{\alpha_m} ALG(J')$. In summary, we have $ALG(J) \leq 2^{\alpha_m} OPT^A(J)$. $\square$

Let ALG_C be the total energy cost of the algorithm in the time interval $[C, C + \frac{y}{2})$, and let OPT^A_C be the total energy cost of an optimal AVR schedule in the time interval $[C, C + \frac{y}{2})$.

For the schedule of Algorithm 7, let A_C be the total energy cost incurred during the time period $[C, C+\frac{y}{2})$ for jobs that are released in the time interval $[C-\frac{y}{2}, C)$. Let $K_{C,i}$ be the set of jobs that are released in $[C - \frac{y}{2}, C)$ and assigned to P_i by the algorithm. Note that we have $A_C = \frac{y}{2} \sum_{i=1}^{m} (\sum_{J_j \in K_{C,i}} \delta_j)^{\alpha_i}$. Furthermore, by Lemma 6.6 we have that $A_C \leq 2^{\alpha_m} OPT^A_C$.

As all jobs have interval length in $[y, 2y]$, the jobs that are executed by the algorithm at some point in the time period $[C, C + \frac{y}{2})$ are released in one of the five intervals $[C - 2y, C - \frac{3y}{2})$, $[C - \frac{3y}{2}, C - y)$, $[C - y, C - \frac{y}{2})$, $[C - \frac{y}{2}, C)$, or $[C, C + \frac{y}{2})$. Therefore, the jobs allocated to processor P_i whose interval overlaps $[C, C + \frac{y}{2})$ are all contained in $U_{C,i} = K_{C-\frac{3y}{2},i} \cup K_{C-y,i} \cup K_{C-\frac{y}{2},i} \cup K_{C,i} \cup K_{C+\frac{y}{2},i}$, and the speed of the processor P_i in the interval $[C + \frac{y}{2}, C + y)$ is at most

$\sum_{J_j \in U_{C,i}} \delta_j$. Therefore, we have :

$$\begin{aligned}
ALG_C &\leq \frac{y}{2}\sum_{i=1}^{m}(\sum_{J_j\in U_{C,i}} \delta_j)^{\alpha_i} \\
&\leq \frac{y}{2}\sum_{i=1}^{m}(\sum_{J_j\in K_{C-\frac{3y}{2},i}} \delta_j + \cdots + \sum_{J_j\in K_{C+\frac{y}{2},i}} \delta_j)^{\alpha_i} \\
&\leq \frac{y}{2}\sum_{i=1}^{m}(5\max\{\sum_{J_j\in K_{C-\frac{3y}{2},i}} \delta_j, \ldots, \sum_{J_j\in K_{C+\frac{y}{2},i}} \delta_j\})^{\alpha_i} \\
&\leq \frac{y}{2}5^{\alpha_m}\sum_{i=1}^{m}\left(\max\{(\sum_{J_j\in K_{C-\frac{3y}{2},i}} \delta_j)^{\alpha_i}, \ldots, (\sum_{J_j\in K_{C+\frac{y}{2},i}} \delta_j)^{\alpha_i}\}\right) \\
&\leq \frac{y}{2}5^{\alpha_m}\sum_{i=1}^{m}\left((\sum_{J_j\in K_{C-\frac{3y}{2},i}} \delta_j)^{\alpha_i} + \cdots + (\sum_{J_j\in K_{C+\frac{y}{2},i}} \delta_j)^{\alpha_i}\right) \\
&\leq 5^{\alpha_m}(A_{C-\frac{3y}{2}} + A_{C-y} + A_{C-\frac{y}{2}} + A_C + A_{C+\frac{y}{2}})
\end{aligned}$$

The total energy cost of Algorithm 7 can then be bounded as follows, where the sums over $C \geq 0$ are to be interpreted as sums over multiples of $\frac{y}{2}$, i.e., over the values $C = u\frac{y}{2}$ for all non-negative integers u:

$$\begin{aligned}
ALG &= \sum_{C\geq 0} ALG_C \\
&\leq \sum_{C\geq 0} 5^{\alpha_m}(A_{C-\frac{3y}{2}} + A_{C-y} + A_{C-\frac{y}{2}} + A_C + A_{C+\frac{y}{2}}) \\
&\leq 5^{\alpha_m}\sum_{C\geq 0}(A_{C-\frac{3y}{2}} + A_{C-y} + A_{C-\frac{y}{2}} + A_C + A_{C+\frac{y}{2}}) \\
&\leq 5^{\alpha_m+1}\sum_{C\geq 0} A_C \\
&\leq 5^{\alpha_m+1}\sum_{C\geq 0} 2^{\alpha_m} OPT_C^A \\
&= 5^{\alpha_m+1}2^{\alpha_m} OPT^A \\
&\leq 5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)OPT
\end{aligned}$$

Here, the fourth inequality follows from Lemma 6.6 and the last inequality holds by Corollary 6.2. Thus, we get the following theorem.

Theorem 6.7. *Algorithm 7 is* $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$*-competitive for jobs with agreeable deadlines and density ratio at most two and interval-length ratio at most two.*

6.4 Arbitrary Interval Lengths and Densities

In this section, we consider jobs with arbitrary interval lengths and densities, we only require that the jobs have agreeable deadlines. Recall that D denotes the density ratio and T the interval-length ratio. Let $\Delta = \max_j \delta_j$ denote the maximum job density, and let $\Lambda = \max_j(d_j - r_j)$ be the maximum interval length. For ease of presentation, we assume that the algorithm knows Δ and Λ, but we will mention later how the algorithm can be adapted to work without this assumption.

The interval lengths of all jobs are in $[\Lambda/T, \Lambda]$ and their densities are in $[\Delta/D, \Delta]$. We classify the jobs into groups such that within each group the interval lengths and densities vary only within a factor of two. Then, each group is scheduled independently of the others using a separate copy of Algorithm 7, but of course all the jobs run on the same set of processors. A job is classified into group $g_{t,d}$ if its interval length is in $[\Lambda/2^t, \Lambda/2^{t-1}]$ and its density in $[\Delta/2^d, \Delta/2^{d-1}]$, where $t \in \{1, \dots, \lceil \log_2 T \rceil\}$ and $d \in \{1, \dots, \lceil \log_2 D \rceil\}$. Jobs that lie at group boundaries can be allocated to one of the two relevant groups arbitrarily. The algorithm for assigning jobs to processors is shown as pseudo-code in Algorithm 8.

Let $\ell(g_{t,d}, i, t')$ be the load (sum of densities of active jobs) of group $g_{t,d}$ on processor P_i at time t', and let $A_{g_{t,d}}$ be the total energy cost for group $g_{t,d}$, assuming that it is the only group running. Let H denote the time when the

Algorithm 8 Allocation of jobs with arbitrary interval lengths and densities

```
Run a separate copy of Algorithm 7 for each group $g_{t,d}$.
while not all jobs allocated do
    $J_j \leftarrow$ next job
    determine group $g_{t,d}$ of $J_j$
    pass $J_j$ to the copy of Algorithm 7 for group $g_{t,d}$
end
```

schedule ends, i.e., the deadline of the last job, and let $OPT(g_{t,d})$ denote the energy cost of the optimal schedule for $g_{t,d}$. We have:

$$A_{g_{t,d}} = \sum_{i=1}^{m} \int_0^H (\ell(g_{t,d}, i, t'))^{\alpha_i} dt' \tag{6.1}$$

From Theorem 6.7, we get:

$$A_{g_{t,d}} \leq 5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1) OPT(g_{t,d}) \tag{6.2}$$

The number of groups used by Algorithm 8 is bounded by $\lceil \log_2 T \rceil \lceil \log_2 D \rceil$. Let ALG be the total energy cost of the algorithm. We can bound ALG as follows, where the sums over t, d (or maxima taken over t, d) are to be understood as sums (or maxima) over all pairs t, d with $1 \leq t \leq \lceil \log_2 T \rceil$ and $1 \leq d \leq \lceil \log_2 D \rceil$:

$$
\begin{aligned}
ALG &= \sum_{i=1}^{m} \int_0^H \Big(\sum_{t,d} \ell(g_{t,d}, i, t') \Big)^{\alpha_i} dt' \\
&\leq \sum_{i=1}^{m} \int_0^H \Big(\lceil \log T \rceil \lceil \log D \rceil \max_{t,d} \{ \ell(g_{t,d}, i, t') \} \Big)^{\alpha_i} dt' \\
&= \sum_{i=1}^{m} \int_0^H \lceil \log T \rceil^{\alpha_i} \lceil \log D \rceil^{\alpha_i} \Big(\max_{t,d} \{ \ell(g_{t,d}, i, t') \} \Big)^{\alpha_i} dt' \\
&\leq \sum_{i=1}^{m} \int_0^H \lceil \log T \rceil^{\alpha_i} \lceil \log D \rceil^{\alpha_i} \Big(\sum_{t,d} (\ell(g_{t,d}, i, t'))^{\alpha_i} \Big) dt' \\
&= \sum_{t,d} \sum_{i=1}^{m} \int_0^H \lceil \log T \rceil^{\alpha_i} \lceil \log D \rceil^{\alpha_i} (\ell(g_{t,d}, i, t'))^{\alpha_i} dt' \\
&\leq \lceil \log T \rceil^{\alpha_m} \lceil \log D \rceil^{\alpha_m} \sum_{t,d} \sum_{i=1}^{m} \int_0^H \Big(\ell(g_{t,d}, i, t') \Big)^{\alpha_i} dt' \\
&\leq \lceil \log T \rceil^{\alpha_m} \lceil \log D \rceil^{\alpha_m} \sum_{t,d} 5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1) OPT(g_{t,d}) \\
&\leq \lceil \log T \rceil^{\alpha_m+1} \lceil \log D \rceil^{\alpha_m+1} 5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m - 1} + 1) OPT
\end{aligned}
$$

Here, the fourth inequality follows from (6.1) and (6.2), and the last inequality follows from the fact that $OPT(g_{t,d}) \leq OPT$ and therefore $\sum_{g_{t,d}} OPT(g_{t,d}) \leq \lceil \log T \rceil \lceil \log D \rceil OPT$. Thus, we get:

Theorem 6.8. *For non-preemptive minimum energy scheduling of jobs with agreeable deadlines on heterogeneous processors, the competitive ratio of Algorithm 8 is at most* $5^{\alpha_m+1} 2^{\alpha_m} (\alpha_m^{\alpha_m} 2^{\alpha_m-1} + 1) \lceil \log D \rceil^{\alpha_m+1} \lceil \log T \rceil^{\alpha_m+1}$.

In case Λ and Δ are not known in advance, we adapt the algorithm as follows. Let ℓ_0 and δ_0 be the interval length and density, respectively, of the job that arrives first. The algorithm stores these values. A job with interval length ℓ and density δ is then assigned to group $g_{t,d}$ with $t = \lceil \log_2(\ell/\ell_0) \rceil$ and $d = \lceil \log_2(\delta/\delta_0) \rceil$, and passed to the copy of Algorithm 7 for that group. If the arriving job is the first of its group $g_{t,d}$, a new copy of Algorithm 7 is started for that group. The number of groups is now bounded by $(\lceil \log_2 T \rceil + 1)(\lceil \log_2 D \rceil + 1)$. The resulting competitive ratio is $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1} + 1)(\lceil \log D \rceil + 1)^{\alpha_m+1}(\lceil \log T \rceil + 1)^{\alpha_m+1}$.

6.5 Summary

In this chapter, we have investigated non-preemptive online scheduling of jobs with agreeable deadlines on heterogeneous speed-scalable processors. We have first shown that the problem can be solved optimally by a simple greedy algorithm if all jobs are identical (i.e., have the same release time, deadline, and work). NAVR is used to schedule the jobs that have been allocated to a processor non-preemptively. From this we have obtained an algorithm with a competitive ratio of at most $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1} + 1)$, for the case where the densities of the jobs differ only within a factor of two and the same holds for their interval lengths. For jobs with equal interval lengths and equal densities, the competitive ratio improves to $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1} + 1)$.

For arbitrary jobs with agreeable deadlines, we have presented an algorithm that classifies the jobs based on density and interval length and allocates the jobs in each class to processors by selecting the processor with the smallest energy cost increase. Our algorithm achieves a competitive ratio of at most $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1} + 1)\lceil \log D \rceil^{\alpha_m+1}\lceil \log T \rceil^{\alpha_m+1}$ for job instances with agreeable deadlines, density ratio D, and interval-length ratio T. This bound is independent of the number of processors. We believe that it is possible to improve the competitive ratio by modifying the algorithm or tightening the analysis.

Chapter 7

Conclusion and Future Work

This thesis has explored a deadline-based scheduling problem of executing HPC-applications by a decentralised multi-cloud system for energy optimisation. It is motivated by the fact that a cloud data-centre consumes a huge amount of energy – estimated to be equivalent to what 25,000 households would consume [72] – that is expected to increase significantly in the near future, and therefore, maintaining that cloud computing being environmentally sustainable is quite significant. For a wider understanding of our considered scheduling problem, this thesis has explored the feasibility of optimising energy consumption from two different perspectives:

- a technical perspective presented in three chapters (3, 4, and 5) based on our developed multi-cloud simulation tool MCST for evaluation and deep insights into outcomes; and
- a theoretical perspective, presented in Chapter 6, that relies on the use of competitive analysis to measure the produced energy cost of the considered scheduling problem (i.e. *minimum energy scheduling*). The problem here represents a simplified form of the energy-aware scheduling problem in multi-cloud systems.

Our exploration commenced with experimentally investigating two different scheduling approaches for executing HPC-applications by a set of clouds, participating in a federated approach, that are: the best-effort (presented in Chapter 4) and the advance-reservation (presented in Chapter 5) approaches. The goal of these proposed approaches was not only minimising the overall energy consumption, but considering along with it, the possibility of maximising resource reliability by mitigating the occurrence of application violations/rejections. Both of these approaches attempt to exploit the potential of sharing distributed cloud resources worldwide as well as the elasticity of dynamically adjusting voltage and frequency of processors (DVFS) for energy optimisation in two scheduling levels: a global level (i.e., between participating heterogeneous clouds) for finding the cloud that gives the best offer, and a local level in each cloud (i.e., between the cloud's homogeneous resources) for mapping an application's tasks to machines (i.e., CPUs). In relation to the global scheduling level, we studied two strategies for choosing the best available resources, offered by a set of clouds, that are the preference and combination rate strategies (PRS and CRS). These strategies aim to optimise energy consumption in general, however, the latter strategy allows us to combine a conflicting aim with energy minimisation, as it also attempts to maximise the reliability of cloud resources.

The evaluation sections of the suggested scheduling approaches (presented in Chapter 4 and Chapter 5) have been primarily devoted to:

- comparing our scheduling methods with the existing scheduling algorithm CMMS [84] that aims to select a cloud that offers the minimal earliest finish time;
- measuring the amount of energy that one could potentially reduce compared with the upper bound of the topmost energy consumption when all clouds run at the highest performance (i.e., under the highest CPU frequencies);

- understanding the effect of the proposed strategies, PRS and CRS, on (1) the average reduction of energy, (2) the number of application rejections/violations, and (3) the utilisation of cloud resources over time; and
- giving a detailed comparison between the proposed scheduling approaches, i.e., the best-effort vs. the advance-reservation regarding several aspects, such as energy optimisation, scheduling efficiency, network utilisation, and the effect of extending application deadlines on the rate of energy reduction.

The second concern in this thesis, introduced in Chapter 6, was to explore and perform the theoretical analysis of the proposed energy-aware online scheduling problem, which can help to deeply understand the outcomes from a different aspect. In fact, the problem of scheduling HPC-applications in multi-cloud systems is naturally very difficult, due to the complexity of online scheduling, particularly when combining two conflicting objectives of minimising energy consumptions and application rejections, and/or when handling various constraints such as application deadlines and tasks dependencies. As such problems would be even harder when considered theoretically, we have, therefore, considered a simplified form of the problem that captures some of the main aspects to draw an analogy with our energy-aware scheduling problem in multi-cloud systems. In a nutshell, we have analysed a non-preemptive online scheduling problem for allocating independent tasks with deadline constraints to heterogeneous processors. This problem fundamentally represents a non-preemptive version of the classical speed-scaling scheduling problem that was studied by Yao et al. [118] but on parallel processors.

In the remainder of this chapter, we conclude the thesis by presenting a brief summary of our achievements, followed by a list of research limitations that we are aware of. Then, we finally end this thesis by giving some possible research directions in relation to our main research objectives.

7.1 Summary of Contributions and Achievements

The major contribution of this thesis to the scope of energy-aware scheduling of cloud applications has been in proposing some possible solutions that have been examined practically and theoretically. The achieved results from the proposed scheduling solutions (some of them have been published in [6, 5, 7]) evidenced that it is possible to reduce a considerable amount of energy while carefully scheduling cloud applications over a multi-cloud system. These scheduling solutions obviously have particular importance in relation with the main aim of green cloud computing for the necessity of increasing energy efficiency.

The technical part of this thesis – it covers the experimental aspects of the two scheduling modes that are the best-effort and the advance-reservation – has focused on combining two energy dimensions while scheduling HPC applications (i.e., energy consumed for execution and data transmission). It has considered simultaneously minimising application rejections and deadline violations, to support resource reliability, with energy optimisation. The practical outcomes of this part are concisely summarised below.

- Approaches to the scheduling of HPC-applications with the goal of energy optimisation should not focus on just the single parameter of energy consumption but incorporate different parameters, ranging from CPU usage levels to data transmissions at network level. In multi-cloud systems, optimal scheduling for energy efficiency that relies on resource utilisation needs to pay special attention to resource reliability.
- The energy consumption of multi-cloud systems that focus on providing high performance services for executing HPC-applications can be efficiently reduced without affecting the desired performance.

- Scheduling an application to a cloud that appears better at the submission time may not lead to the best energy saving result over time. In particular, balancing the workloads among all clouds is sometimes important, which helps to avoid some cases where applications scheduled later need to be forcibly allocated to less efficient clouds.

In the theoretical part of this thesis, we have presented the first online algorithms for the non-preemptive scheduling of jobs with agreeable deadlines on heterogeneous parallel processors, published in [7]. We have used NAVR to schedule the jobs that have been allocated to a processor non-preemptively. The performed analysis and the eventual outcomes, achieved from this part, are concisely summarised below.

- Initially, we have explained that the problem can be solved optimally by a simple greedy algorithm if all jobs are identical (i.e., have the same release time, deadline, and work). From this we have obtained an algorithm with a competitive ratio of at most $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$, for the case where the densities of the jobs differ only within a factor of two and the same holds for their interval lengths.
- For agreeable jobs with equal interval lengths and equal densities, the competitive ratio improves to $3^{\alpha_m+1}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)$.
- For arbitrary jobs with agreeable deadlines, we have presented an algorithm that classifies the jobs based on density and interval length and allocates the jobs in each class to processors by selecting the processor with the smallest energy cost increase. Our algorithm achieves a competitive ratio of at most $5^{\alpha_m+1}2^{\alpha_m}(\alpha_m^{\alpha_m}2^{\alpha_m-1}+1)\lceil\log D\rceil^{\alpha_m+1}\lceil\log T\rceil^{\alpha_m+1}$ for job instances with agreeable deadlines, density ratio D, and interval-length ratio T. This bound is independent of the number of processors.

We believe that it is possible to improve the competitive ratio by modifying the algorithm or tightening the analysis.

7.2 Limitations and Assumptions

While our research problems, explained in Section 1.2, have been experimentally tackled, including a theoretical study of a simplified form of the problems, the proposed solutions and assumptions have two general limitations that we are aware of, reported concisely as follows.

First of all, our designed prototype simulation tool MCST is too abstract to represent the underlying characteristics of a real multi-cloud infrastructure that should support a typical interconnection network model (i.e., precisely simulate communication between geographically distributed clouds, and also communication within cloud data-centres). Therefore, the overall energy cost achieved in this thesis may not be quite realistic as some important energy parameters were not considered, such as the energy cost for activating idle servers. However, to simplify our simulation tool for gaining clear insights, we had to abstract away many unnecessary characteristics that seem closely related to the cloud infrastructures (e.g., memory or services), but unimportant in the context of this thesis.

Second of all, although the calculation of energy consumption of data transmissions through the Internet is a major challenge (as explained in Section 4.1.3) and no accurate solution is available, to our knowledge, we have made a rational assumption to linearly estimate the energy cost of a transmission over a given distance based on a typical transmission using a fixed rate of $\mu = 0.2kWh/GB$ (obtained from [44]) over a distance of 500 km. Nevertheless, if a better solution for calculating transmission energy costs is obtainable that fits our scenarios, it can be straightforwardly implemented into our simulation tool MCST. In addition, our assumption here is limited to covering transmission energy costs between distributed clouds, but not within the local interconnection network of cloud resources. However, since we assume that each cloud has a single data-centre site, the energy consumption of data transmissions from one server to another through the local network should be negligible compared to the energy consumption of the servers and of data transmissions between data-centres.

7.3 Future Work

There still remain many areas and directions for achieving further energy optimisations by intelligently scheduling HPC-applications over distributed multi-cloud systems. We discuss below our main interesting future work that can be organised into two categorises: (1) improving our system model in a way that would address some of the limitations, reported in Section 7.2, and (2) inventing more features as supplements to the best-effort/advance-reservation scheduling solutions in addition to improving our theoretical approach. This future work, apart from providing new insights into theoretical aspects, would generally promote the suggested approaches of this thesis not only for energy efficiency, but also for enhancing the reliability of cloud resources.

7.3.1 Extension of the Considered System Model

According to the main limitations of the suggested scheduling approaches in this thesis, reported in Section 7.2, extending our system model (discussed in Section 3.1) to consider a more realistic multi-cloud environment is highly recommended for future work. This would help to cover more real-world applications and to see how various scenarios can affect the energy efficiency. A typical extension of the current system model should ideally support handling the complex requirements of modelling: (1) SLAs, (2) cloud applications, and (3) the interconnection network within a cloud data-centre.

In general, future research can concentrate on figuring out more accurately the impact of scheduling different types of real cloud applications under different requirements (mainly, dependent vs. independent tasks and preemptive vs. non-preemptive tasks) on energy efficiency. A possible categorisation of application workloads, presented in [79], can be adopted which depends broadly on the size of computational execution as well as the size of required data transmission, as follows.

- *Computationally Intensive Workloads*: This category includes applications that need a high amount of energy for computational activities (e.g., HPC-applications), but a low amount of energy for data transmission between cloud resources in both networks (the global between clouds and the local between cloud resources).
- *Data-Intensive Workloads*: This category focuses on applications that do not require high computational activities, but usually utilise the network heavily for transmitting application datasets. A typical example of this kind of applications can be seen in online large video streaming.
- *Balanced Workloads*: This category covers applications that need a high amount of energy to execute cloud applications as well as to transmit their datasets.

As each application from this categorisation would obviously need a particular energy saving technique to ideally optimise the overall energy consumption, this may highlight the importance of applying a specific scheduling solution according to the application type.

Another interesting direction for future research is to empirically evaluate our proposed algorithms by implementing them in a federated system testbed like

Open Cirrus [92]. This testbed consists of loosely-coupled distributed sites each of which is owned and managed by a different participant (e.g., HP, Intel, Yahoo, etc.). The objective could be to understand the behaviours of the main critical parameters that are mostly in relation with wasting energy and/or raising the energy consumption unnecessarily.

7.3.2 Elaborating and Enriching our Scheduling Approaches

Our evaluations in Sections 4.4 and 5.3 have revealed that there is an interdependency between using one of the proposed strategies (i.e., PRS and CRS) for scheduling decisions and the characteristics of the submitted applications under a certain level of resource occupation. As a consequence, we intend to further optimise energy consumption by designing an adaptive algorithm that can dynamically adjust the decision strategy (if needed) based on the given scheduling problem.

In Chapter 4, the proposed best-effort approach makes scheduling decisions by assessing cloud offers that have no guarantee for their available resources. This bold decision usually results in violating the execution of some already scheduled applications. Thus, it would be interesting to develop a smart predictor (e.g., using a data mining approach) that helps in enhancing the precision of the best-effort scheduling decision so as to predict whether a submitted application would be violated or not in advance. Here, the minimisation of application violation cases by such a predictor has two advantages. Firstly, it gives a better chance for applications to get executed by other clouds, and second, it helps in saving wasted energy that would be consumed by partially executing applications before their deadlines get violated.

For the advance-reservation approach, introduced in Chapter 5, a potential investigation of energy efficiency can be carried out. The idea is to design a rescheduling mechanism, relying mainly on the release of resource reservations, that may help in optimising further the energy consumption as well as application rejection cases. To be more precise, further energy reduction here could be gained by expanding the time frame that probably enables a further reduction of the CPU frequencies that have already been set for executing application tasks. Minimising application rejection cases is feasible, because the more scheduled applications are shifted to some earlier possible times (meaning that application executions would finish at some earlier times), the higher the level of resources that are available early, which in turn helps to accept more applications.

Concerning the theoretical approach, introduced in Chapter 6, one possible direction for improving the ratio could be to consider how the method of dispatching jobs can be combined with another scheduler on each processor instead of NAVR (explained in Section 6.2). It would also be interesting to prove lower bounds on the best possible competitive ratio. An open problem is to drop the assumption of agreeable deadlines and consider the online heterogeneous speed scaling problem without preemption for jobs with arbitrary release times and deadlines.

Bibliography

[1] Susanne Albers and Antonios Antoniadis. “Race to idle: New algorithms for speed scaling with a sleep state”. In: *ACM Trans. Algorithms* 10.2 (2014), 9:1–9:31. DOI: 10.1145/2556953. URL: http://doi.acm.org/10.1145/2556953 (cited on page 18).

[2] Susanne Albers, Antonios Antoniadis, and Gero Greiner. “On multi-processor speed scaling with migration”. In: *J. Comput. Syst. Sci.* 81.7 (2015), pages 1194–1209. DOI: 10.1016/j.jcss.2015.03.001. URL: http://dx.doi.org/10.1016/j.jcss.2015.03.001 (cited on pages 38–40).

[3] Susanne Albers, Evripidis Bampis, Dimitrios Letsios, Giorgio Lucarelli, and Richard Stotz. “Scheduling on Power-Heterogeneous Processors”. In: *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings.* 2016, pages 41–54. DOI: 10.1007/978-3-662-49529-2_4. URL: http://dx.doi.org/10.1007/978-3-662-49529-2_4 (cited on pages 39, 40, 124).

[4] Susanne Albers, Fabian Müller, and Swen Schmelzer. “Speed Scaling on Parallel Processors”. In: *Algorithmica* 68.2 (2014), pages 404–425. DOI: 10.1007/s00453-012-9678-7. URL: http://dx.doi.org/10.1007/s00453-012-9678-7 (cited on pages 38, 40).

[5] Aeshah Alsughayyir and Thomas Erlebach. “A Bi-objective Scheduling Approach for Energy Optimisation of Executing and Transmitting HPC Applications in Decentralised Multi-cloud Systems”. In: *Proceedings of*

the 16th International Symposium on Parallel and Distributed Computing, ISPDC 2017, 3-6 July 2017, Innsbruck, Austria. (To appear). 2017 (cited on pages 7, 8, 10, 11, 94, 142).

[6] Aeshah Alsughayyir and Thomas Erlebach. "Energy Aware Scheduling of HPC Tasks in Decentralised Cloud Systems". In: *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016, Heraklion, Crete, Greece, February 17-19, 2016.* 2016, pages 617–621 (cited on pages 7, 8, 10, 11, 60, 142).

[7] Aeshah Alsughayyir and Thomas Erlebach. "Online Algorithms for Non-preemptive Speed Scaling on Power-Heterogeneous Processors". In: *Proceedings of the 11th Annual International Conference on Combinatorial Optimization and Applications, COCOA 2017, December 16-18, 2017, in Shanghai, China. (To appear).* 2017 (cited on pages 8, 10, 11, 121, 142, 143).

[8] *Amazon Web Services.* `http://aws.amazon.com/`. [Online; accessed September-2017] (cited on page 4).

[9] L. Anand, Debasish Ghose, and V. Mani. "ELISA: an estimated load information scheduling algorithm for distributed computing systems". In: *Computers & Mathematics with Applications* 37.8 (1999), pages 57–85 (cited on pages 26, 28).

[10] Antonios Antoniadis and Chien-Chung Huang. "Non-preemptive speed scaling". In: *Journal of Scheduling* 16.4 (2013), pages 385–394 (cited on page 40).

[11] Patricia Arroba, José Manuel Moya, José L. Ayala, and Rajkumar Buyya. "Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers". In: *Concurrency and Computation: Practice and Experience* 29.10 (2017) (cited on page 18).

[12] Arutyun Avetisyan, Roy H. Campbell, Indranil Gupta, Michael T. Heath, Steven Y. Ko, Gregory R. Ganger, Michael A. Kozuch, David R. O'Hallaron, Marcel Kunze, Thomas T. Kwan, Kevin Lai, Martha Lyons, Dejan S. Milojicic, Hing Yan Lee, Yeng Chai Soh, Ng Kwang Ming, Jing-Yuan Luke, and Han Namgoong. "Open Cirrus: A Global Cloud Computing Testbed". In: *IEEE Computer* 43.4 (2010), pages 35–43. DOI: `10.1109/MC.2010.111`. URL: `https://doi.org/10.1109/MC.2010.111` (cited on page 4).

[13] Jayant Baliga, Robert Ayre, Kerry Hinton, and Rodney S. Tucker. "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport". In: *Proceedings of the IEEE* 99.1 (2011), pages 149–167. DOI: `10.1109/JPROC.2010.2060451`. URL: `https://doi.org/10.1109/JPROC.2010.2060451` (cited on page 20).

[14] Evripidis Bampis. "Algorithmic Issues in Energy-Efficient Computation". In: *International Conference on Discrete Optimization and Operations Research*. Springer. 2016, pages 3–14 (cited on pages 25, 38).

[15] Evripidis Bampis, Alexander V. Kononov, Dimitrios Letsios, Giorgio Lucarelli, and Ioannis Nemparis. "From preemptive to non-preemptive speed-scaling scheduling". In: *Discrete Applied Mathematics* 181 (2015), pages 11–20. DOI: `10.1016/j.dam.2014.10.007`. URL: `http://dx.doi.org/10.1016/j.dam.2014.10.007` (cited on pages 38–40, 123).

[16] Evripidis Bampis, Dimitrios Letsios, and Giorgio Lucarelli. "Speed-Scaling with No Preemptions". In: *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*. 2014, pages 259–269. DOI: `10.1007/978-3-319-13075-0_21`. URL: `http://dx.doi.org/10.1007/978-3-319-13075-0_21` (cited on pages 39, 40).

[17] Jeff Barr. *Amazon EC2 Beta*. https://aws.amazon.com/blogs/aws/amazon_ec2_beta/. Blog. [Retrieved 20-Aug-2017]. 2006 (cited on page 13).

[18] Paul C. Bell and Prudence W. H. Wong. "Multiprocessor speed scaling for jobs with arbitrary sizes and deadlines". In: *J. Comb. Optim.* 29.4 (2015), pages 739–749. DOI: 10.1007/s10878-013-9618-8. URL: http://dx.doi.org/10.1007/s10878-013-9618-8 (cited on pages 38, 40).

[19] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Y. Zomaya. "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems". In: *Advances in Computers* 82 (2011), pages 47–111. DOI: 10.1016/B978-0-12-385512-1.00003-7. URL: https://doi.org/10.1016/B978-0-12-385512-1.00003-7 (cited on pages 19, 32).

[20] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. "A survey of design techniques for system-level dynamic power management". In: *IEEE Trans. VLSI Syst.* 8.3 (2000), pages 299–316. DOI: 10.1109/92.845896. URL: https://doi.org/10.1109/92.845896 (cited on page 19).

[21] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, and Monique Morrow. "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability". In: *Fourth International Conference on Internet and Web Applications and Services, ICIW 2009, 24-28 May 2009, Venice/Mestre, Italy.* 2009, pages 328–336. DOI: 10.1109/ICIW.2009.55. URL: https://doi.org/10.1109/ICIW.2009.55 (cited on page 4).

[22] Nik Bessis, Stelios Sotiriadis, Florin Pop, and Valentin Cristea. "Optimizing the Energy Efficiency of Message Exchanging for Service Distribution in Interoperable Infrastructures". In: *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012.* 2012, pages 105–112. DOI:

10.1109/iNCoS.2012.16. URL: https://doi.org/10.1109/iNCoS.2012.16 (cited on page 31).

[23] Ricardo Bianchini and Ramakrishnan Rajamony. “Power and Energy Management for Server Systems”. In: *IEEE Computer* 37.11 (2004), pages 68–74. DOI: 10.1109/MC.2004.217. URL: https://doi.org/10.1109/MC.2004.217 (cited on pages 5, 16, 42).

[24] James Blythe, S. Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, Anirban Mandal, and Ken Kennedy. “Task scheduling strategies for workflow-based applications in grids”. In: *5th International Symposium on Cluster Computing and the Grid (CCGrid 2005), 9-12 May, 2005, Cardiff, UK.* 2005, pages 759–767 (cited on page 57).

[25] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge university press, 2005 (cited on page 24).

[26] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Ladislau Bölöni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, Bin Yao, Debra A. Hensgen, and Richard F. Freund. “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”. In: *J. Parallel Distrib. Comput.* 61.6 (2001), pages 810–837 (cited on page 57).

[27] Peter Brucker. *Scheduling Algorithms.* 5th. Springer-Verlag Berlin Heidelberg, 2007. ISBN: 978-3-540-69516-5 (cited on page 24).

[28] Marc Bux and Ulf Leser. “DynamicCloudSim: Simulating heterogeneity in computational clouds”. In: *Future Generation Comp. Syst.* 46 (2015), pages 85–99. URL: https://doi.org/10.1016/j.future.2014.09.007 (cited on pages 55–57).

[29] Rajkumar Buyya, Anton Beloglazov, and Jemal H. Abawajy. “Energy-Efficient Management of Data Center Resources for Cloud Computing:

A Vision, Architectural Elements, and Open Challenges". In: *CoRR* abs/1006.0308 (2010) (cited on page 5).

[30] Rajkumar Buyya and M. Manzur Murshed. "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing". In: *Concurrency and Computation: Practice and Experience* 14.13-15 (2002), pages 1175–1220 (cited on pages 55, 57, 58).

[31] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. "InterCloud Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services". In: *Algorithms and Architectures for Parallel Processing, 10th International Conference, ICA3PP 2010, Busan, Korea, May 21-23, 2010. Proceedings. Part I.* 2010, pages 13–31. DOI: `10.1007978-3-642-13119-6_2`. URL: `httpsdoi.org10.1007978-3-642-13119-6_2` (cited on page 14).

[32] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". In: *Future Generation Comp. Syst.* 25.6 (2009), pages 599–616. DOI: `10.1016/j.future.2008.12.001`. URL: `https://doi.org/10.1016/j.future.2008.12.001` (cited on page 4).

[33] Rodrigo N. Calheiros, Marco Aurélio Stelmar Netto, César A. F. De Rose, and Rajkumar Buyya. "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of Cloud computing applications". In: *Softw., Pract. Exper.* 43.5 (2013), pages 595–612. URL: `https://doi.org/10.1002/spe.2124` (cited on page 56).

[34] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Softw., Pract. Exper.* 41.1 (2011), pages 23–50 (cited on pages 55, 57).

[35] Louis-Claude Canon and Emmanuel Jeannot. "Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments". In: *IEEE Trans. Parallel Distrib. Syst.* 21.4 (2010), pages 532–546 (cited on page 22).

[36] Fei Cao and Michelle M. Zhu. "Energy-Aware Workflow Job Scheduling for Green Clouds". In: *2013 IEEE International Conference on Green Computing and Communications (GreenCom) and IEEE Internet of Things (iThings) and IEEE Cyber, Physical and Social Computing (CPSCom), Beijing, China, August 20-23, 2013.* 2013, pages 232–239. DOI: `10.1109/GreenCom-iThings-CPSCom.2013.58`. URL: `https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.58` (cited on pages 28, 34, 37).

[37] Henri Casanova. "SimGrid: A Toolkit for the Simulation of Application Scheduling". In: *First IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18, 2001, Brisbane, Australia.* 2001, pages 430–441 (cited on page 55).

[38] Antonio Celesti, Francesco Tusa, Massimo Villari, and Antonio Puliafito. "How to Enhance Cloud Architectures to Enable Cross-Federation". In: *IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5-10 July, 2010.* 2010, pages 337–345. DOI: `10.1109/CLOUD.2010.46`. URL: `https://doi.org/10.1109/CLOUD.2010.46` (cited on page 4).

[39] Weiwei Chen and Ewa Deelman. "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments". In: *8th IEEE International Conference on E-Science, e-Science 2012, Chicago, IL, USA,*

October 8-12, 2012. 2012, pages 1–8. URL: https://doi.org/10.1109/eScience.2012.6404430 (cited on pages 55–57).

[40] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, and Natarajan Gautam. “Managing server energy and operational costs in hosting centers”. In: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada*. 2005, pages 303–314. DOI: 10.1145/1064212.1064253. URL: http://doi.acm.org/10.1145/1064212.1064253 (cited on page 18).

[41] Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, and Natarajan Gautam. “Managing server energy and operational costs in hosting centers”. In: *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2005, June 6-10, 2005, Banff, Alberta, Canada*. 2005, pages 303–314. URL: http://doi.acm.org/10.1145/1064212.1064253 (cited on page 19).

[42] *CloudStack*. https://cloudstack.apache.org/. [Online; accessed September-2017]. 2010 (cited on page 4).

[43] Vincent Cohen-Addad, Zhentao Li, Claire Mathieu, and Ioannis Milis. “Energy-Efficient Algorithms for Non-preemptive Speed-Scaling”. In: *Approximation and Online Algorithms - 12th International Workshop, WAOA 2014, Wrocław, Poland, September 11-12, 2014, Revised Selected Papers*. 2014, pages 107–118. DOI: 10.1007/978-3-319-18263-6_10. URL: http://dx.doi.org/10.1007/978-3-319-18263-6_10 (cited on pages 38, 39).

[44] Vlad C. Coroama and Lorenz M. Hilty. “Assessing Internet energy intensity: A review of methods and results”. In: *Environmental impact assessment review* 45 (2014), pages 63–68 (cited on pages 67, 144).

[45] Vlad C. Coroama, Daniel Schien, Chris Preist, and Lorenz M. Hilty. "The Energy Intensity of the Internet: Home and Access Networks". In: *ICT Innovations for Sustainability*. 2015, pages 137–155. DOI: 10.1007/978-3-319-09228-7_8. URL: https://doi.org/10.1007/978-3-319-09228-7_8 (cited on page 67).

[46] Rachel Courtland. "The end of the shrink". In: *IEEE Spectrum* 50.11 (2013), pages 26–29. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2013.6655835 (cited on page 15).

[47] D. Dharwar, S. S. Bhat, V. Srinivasan, D. Sarma, and P. K. Banerjee. "Approaches Towards Energy-Efficiency in the Cloud for Emerging Markets". In: *2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. 2012, pages 1–6 (cited on page 18).

[48] Zhihui Du, Rong Ge, Victor W. Lee, Richard W. Vuduc, David A. Bader, and Ligang He. "Modeling the Power Variability of Core Speed Scaling on Homogeneous Multicore Systems". In: *Scientific Programming* 2017 (2017), 8686971:1–8686971:13. DOI: 10.1155/2017/8686971. URL: https://doi.org/10.1155/2017/8686971 (cited on page 65).

[49] Enerdata. *Global Energy Statistical Yearbook 2014*. https://yearbook.enerdata.net/. [Online; accessed 28-September-2017] (cited on page 2).

[50] *Eucalyptus*. http://open.eucalyptus.com/. [Online; accessed September-2017]. 2009 (cited on page 4).

[51] Dror G. Feitelson, Dan Tsafrir, and David Krakov. "Experience with using the Parallel Workloads Archive". In: *J. Parallel Distrib. Comput.* 74.10 (2014), pages 2967–2982. URL: http://dx.doi.org/10.1016/j.jpdc.2014.06.013 (cited on pages 52, 59, 81).

[52] Saurabh Kumar Garg and Rajkumar Buyya. "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations". In: *IEEE 4th International Conference on Utility and Cloud Computing, UCC 2011, Melbourne, Australia, December 5-8, 2011.* 2011, pages 105–113. URL: `https://doi.org/10.1109/UCC.2011.24` (cited on pages 37, 43, 47, 55–57).

[53] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. "Energy-Efficient Scheduling of HPC Applications in Cloud Computing Environments". In: *CoRR* abs/0909.1146 (2009). URL: `http://arxiv.org/abs/0909.1146` (cited on pages 19, 28, 34, 35, 37, 80, 81, 104).

[54] *Gartner's Strategic Technology Trends for 2017.* 2016. URL: `http://www.gartner.com/technology/home.jsp` (cited on page 2).

[55] Neven Abou Gazala. "Power management techniques for conserving energy in multiple system components". PhD thesis. University of Pittsburgh, 2008 (cited on pages 16, 63).

[56] Marco E. T. Gerards, Johann L. Hurink, and Philip K. F. Hölzenspies. "A survey of offline algorithms for energy minimization under deadline constraints". In: *J. Scheduling* 19.1 (2016), pages 3–19. DOI: `10.1007/s10951-015-0463-8`. URL: `http://dx.doi.org/10.1007/s10951-015-0463-8` (cited on pages 25, 38).

[57] Imran Ghani, Naghmeh Niknejad, and Seung Ryul Jeong. "Energy saving in green cloud computing data centers A review". In: *Journal of Theoretical and Applied Information Technology* 74.1 (2015) (cited on page 16).

[58] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Annals of discrete mathematics* 5 (1979), pages 287–326 (cited on page 40).

[59] Gero Greiner, Tim Nonner, and Alexander Souza. “The Bell Is Ringing in Speed-Scaled Multiprocessor Scheduling”. In: *Theory Comput. Syst.* 54.1 (2014), pages 24–44. DOI: `10.1007/s00224-013-9477-9`. URL: `https://doi.org/10.1007/s00224-013-9477-9` (cited on pages 38, 40).

[60] Jordi Guitart. “Toward sustainable data centers: A comprehensive energy management strategy”. In: *Computing* 99.6 (2017), pages 597–615. DOI: `10.1007/s00607-016-0501-1`. URL: `https://doi.org/10.1007/s00607-016-0501-1` (cited on page 26).

[61] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. “Scalably Scheduling Power-Heterogeneous Processors”. In: *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I.* 2010, pages 312–323. DOI: `10.1007/978-3-642-14165-2_27`. URL: `https://doi.org/10.1007/978-3-642-14165-2_27` (cited on page 38).

[62] Willy Herroelen and Roel Leus. “Robust and reactive project scheduling: a review and classification of procedures”. In: *International Journal of Production Research* 42.8 (2004), pages 1599–1620 (cited on page 22).

[63] Fred Howell and Ross Mcnab. “simjava: A Discrete Event Simulation Library For Java”. In: *In International Conference on Web-Based Modeling and Simulation.* 1998, pages 51–56 (cited on pages 47, 50, 55, 56, 58).

[64] He Huang and Liqiang Wang. “P&P: A Combined Push-Pull Model for Resource Monitoring in Cloud Computing Environment”. In: *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on.* 2010, pages 260–267. DOI: `10.1109/CLOUD.2010.85` (cited on pages 31, 63).

[65] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. "Energy efficient DVFS scheduling for mixed-criticality systems". In: *2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, October 12-17, 2014*. 2014, 11:1–11:10. URL: `http://doi.acm.org/10.1145/2656045.2656057` (cited on pages 3, 26).

[66] Ye Huang, Nik Bessis, Peter Norrington, Pierre Kuonen, and Béat Hirsbrunner. "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm". In: *Future Generation Comp. Syst.* 29.1 (2013), pages 402–415. DOI: `10.1016/j.future.2011.05.006`. URL: `https://doi.org/10.1016/j.future.2011.05.006` (cited on pages 14, 26, 28, 29, 37).

[67] Sandy Irani and Kirk Pruhs. "Algorithmic problems in power management". In: *SIGACT News* 36.2 (2005), pages 63–76. DOI: `10.1145/1067309.1067324`. URL: `http://doi.acm.org/10.1145/1067309.1067324` (cited on pages 17, 19).

[68] Santiago Iturriaga, Sergio Nesmachnow, Andrei Tchernykh, and Bernabé Dorronsoro. "Multiobjective Workflow Scheduling in a Federation of Heterogeneous Green-Powered Data Centers". In: *IEEE/ACM 16th International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2016, Cartagena, Colombia, May 16-19, 2016*. 2016, pages 596–599. DOI: `10.1109/CCGrid.2016.34`. URL: `https://doi.org/10.1109/CCGrid.2016.34` (cited on pages 26, 28, 32, 37).

[69] Joti Lal Jain, Sri Gopal Mohanty, and Walter Böhm. *A course on queueing models*. New York: Chapman and Hall/CRC., 2016 (cited on page 36).

[70] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991. ISBN: 978-0-471-50336-1 (cited on page 35).

[71] Fredy Juarez, Jorge Ejarque, and Rosa M Badia. “Dynamic energy-aware scheduling for parallel task-based application in cloud computing”. In: *Future Generation Computer Systems* (2016) (cited on pages 26, 28, 33, 37).

[72] James M. Kaplan, William Forrest, and Noah Kindler. *Revolutionizing Data Center Energy Efficiency, McKinsey & Company, July 2009.* `https://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf`. [Online; accessed September-2017] (cited on page 139).

[73] Tarandeep Kaur and Inderveer Chana. “Energy Efficiency Techniques in Cloud Computing: A Survey and Taxonomy”. In: *ACM Comput. Surv.* 48.2 (2015), 22:1–22:46. DOI: `10.1145/2742488`. URL: `http://doi.acm.org/10.1145/2742488` (cited on pages 3, 18, 25).

[74] Gábor Kecskeméti, Attila Kertész, Attila Marosi, and Zsolt Németh. “Strategies for Increased Energy Awareness in Cloud Federations”. In: *High-performance Computing on Complex Environments.* Edited by Emmanuel Jeannot and Julius Zilinskas. John Wiley & Sons, 2013 (cited on page 32).

[75] Attila Kertész. “Characterizing Cloud Federation Approaches”. In: *Cloud Computing - Challenges, Limitations and R&D Solutions.* Edited by Zaigham Mahmood. Springer International Publishing, Oct. 2014, pages 277–296. ISBN: 978-3-319-10529-1. DOI: `10.1007/978-3-319-10530-7_12`. URL: `https://doi.org/10.1109/CLOUD.2017.93` (cited on page 4).

[76] Yacine Kessaci, Nouredine Melab, and El-Ghazali Talbi. “A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation”. In: *Cluster Computing* 16.3 (2013), pages 451–468. DOI: `10.1007/s10586-012-0210-2`. URL: `https://doi.org/10.1007/s10586-012-0210-2` (cited on page 32).

[77] Minjoong Kim, Yoondeo k Ju, Jinseok Chae, and Moonju Park. "A Simple Model for Estimating Power Consumption of a Multicore Server System". In: *International Journal of Multimedia and Ubiquitous Engineering* 9.2 (2014), pages 153–160. ISSN: 1975-0080 (cited on page 65).

[78] John Leslie King. "Centralized versus Decentralized Computing: Organizational Considerations and Management Options". In: *ACM Comput. Surv.* 15.4 (1983), pages 319–349. DOI: `10.1145/289.290`. URL: `http://doi.acm.org/10.1145/289.290` (cited on page 14).

[79] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers". In: *The Journal of Supercomputing* 62.3 (2012), pages 1263–1283. URL: `https://doi.org/10.1007/s11227-010-0504-1` (cited on pages 55–58, 146).

[80] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai, and Marcel Kunze. "Cloud federation". In: *CLOUD COMPUTING* 2011 (2011), pages 32–38 (cited on page 4).

[81] Tak-Wah Lam, Lap-Kei Lee, Isaac KK To, and Prudence WH Wong. "Speed scaling functions for flow time scheduling based on active job count". In: *European Symposium on Algorithms.* Springer. 2008, pages 647–659 (cited on page 38).

[82] Gregor von Laszewski, Lizhe Wang, Andrew J. Younge, and Xi He. "Power-aware scheduling of virtual machines in DVFS-enabled clusters". In: *Proceedings of the 2009 IEEE International Conference on Cluster Computing, August 31 - September 4, 2009, New Orleans, Louisiana, USA*. 2009, pages 1–10. DOI: `10.1109/CLUSTR.2009.5289182`. URL: `https://doi.org/10.1109/CLUSTR.2009.5289182` (cited on page 65).

[83] Etienne Le Sueur and Gernot Heiser. “Dynamic Voltage and Frequency Scaling: The Laws of Diminishing Returns”. In: *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*. HotPower’10. Vancouver, BC, Canada: USENIX Association, 2010, pages 1–8 (cited on page 18).

[84] Jiayin Li, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin, and Zonghua Gu. “Online optimization for scheduling preemptable tasks on IaaS cloud systems”. In: *J. Parallel Distrib. Comput.* 72.5 (2012), pages 666–677. DOI: 10.1016/j.jpdc.2012.02.002. URL: https://doi.org/10.1016/j.jpdc.2012.02.002 (cited on pages 8, 9, 14, 26, 28, 29, 31, 37, 79, 81, 103, 118, 140).

[85] Minming Li, Andrew C Yao, and Frances F Yao. “Discrete and continuous min-energy schedules for variable voltage processors”. In: *Proceedings of the National Academy of Sciences of the United States of America* 103.11 (2006), pages 3983–3987 (cited on page 17).

[86] Zukui Li and Marianthi G. Ierapetritou. “Process scheduling under uncertainty: Review and challenges”. In: *Computers & Chemical Engineering* 32.4-5 (2008), pages 715–727 (cited on pages 23, 71).

[87] Jacob Rubin Lorch. “Operating systems techniques for reducing processor energy consumption”. PhD thesis. University of California, Berkeley, 2001 (cited on page 16).

[88] Andrea Margheri, Md. Sadek Ferdous, Mu Yang, and Vladimiro Sassone. “A Distributed Infrastructure for Democratic Cloud Federations”. In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, June 25-30, 2017*. 2017, pages 688–691. DOI: 10.1109/CLOUD.2017.93. URL: https://doi.org/10.1109/CLOUD.2017.93 (cited on page 4).

[89] Attila Marosi, Gabor Kecskemeti, Attila Kertesz, and Peter Kacsuk. "FCM: An architecture for integrating IaaS cloud systems". In: *In Proceedings of the Second International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2011), IARIA,Rome, Italy.* 2011, pages 7–12 (cited on page 4).

[90] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V. Vasilakos. "Cloud Computing: Survey on Energy Efficiency". In: *ACM Comput. Surv.* 47.2 (2014), 33:1–33:36. DOI: `10.1145/2656204`. URL: `https://doi.org/10.1145/2656204` (cited on pages 2, 25).

[91] *Nimbus.* `http://www.nimbusproject.org/`. [Online; accessed September-2017]. 2009 (cited on page 4).

[92] *Open Cirrus.* `http://www.opencirrus.org/`. [Online; accessed September-2017] (cited on pages 4, 147).

[93] *OpenNebula.* `https://opennebula.org/`. [Online; accessed September-2017]. 2002 (cited on page 4).

[94] Anne-Cécile Orgerie and Laurent Lefèvre. "Energy-Efficient Reservation Infrastructure for Grids, Clouds and Networks". In: *Energy Efficient Distributed Computing Systems* (2012), pages 133–162 (cited on page 26).

[95] *Parallel Workloads Archive.* `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`. [Online; accessed 14-May-2017]. 2005 (cited on pages 52, 59, 81).

[96] Ilia Pietri and Rizos Sakellariou. "Mapping Virtual Machines onto Physical Machines in Cloud Computing: A Survey". In: *ACM Comput. Surv.* 49.3 (2016), 49:1–49:30. DOI: `10.1145/2983575`. URL: `http://doi.acm.org/10.1145/2983575` (cited on page 25).

[97] Michael L Pinedo. *Scheduling: theory, algorithms, and systems.* Springer, 2016 (cited on page 24).

[98] Deepak Poola, Saurabh Kumar Garg, Rajkumar Buyya, Yun Yang, and Kotagiri Ramamohanarao. "Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds". In: *28th IEEE International Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, BC, Canada, May 13-16, 2014*. 2014, pages 858–865 (cited on page 22).

[99] Muhammad Imran Qureshi, Amran Md. Rasli, and Khalid Zaman. "Energy crisis, greenhouse gas emissions and sectoral growth reforms: repairing the fabricated mosaic". In: *Journal of Cleaner Production* 112.Part 5 (2016), pages 3657 –3666. ISSN: 0959-6526. URL: `http://www.sciencedirect.com/science/article/pii/S0959652615011063` (cited on page 2).

[100] Antonio Regalado. *Who Coined 'Cloud Computing'?, (October 31, 2011).* `https://www.technologyreview.com/s/425970/who-coined-cloud-computing/`. [Online; accessed 28-September-2017] (cited on page 1).

[101] Ivan Rodero, Juan Jaramillo, Andres Quiroz, Manish Parashar, Francesc Guim, and Stephen Poole. "Energy-efficient application-aware online provisioning for virtualized clouds and data centers". In: *International Green Computing Conference 2010, Chicago, IL, USA, 15-18 August 2010.* 2010, pages 31–45 (cited on page 3).

[102] Ruchir Shah, Bharadwaj Veeravalli, and Manoj Misra. "On the Design of Adaptive and Decentralized Load Balancing Algorithms with Load Estimation for Computational Grid Environments". In: *IEEE Trans. Parallel Distrib. Syst.* 18.12 (2007), pages 1675–1686. DOI: `10.1109/TPDS.2007.1115`. URL: `https://doi.org/10.1109/TPDS.2007.1115` (cited on pages 26, 28, 29).

[103] Yogesh Sharma, Bahman Javadi, Weisheng Si, and Daniel Sun. "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy". In: *J. Network and Computer Applications* 74 (2016), pages 66–85 (cited on page 3).

[104] Stelios Sotiriadis, Nik Bessis, and Nick Antonopoulos. “Towards Inter-cloud Schedulers: A Survey of Meta-scheduling Approaches”. In: *2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2011, Barcelona, Catalonia, Spain, October 26-28, 2011.* 2011, pages 59–66. DOI: `10.1109/3PGCIC.2011.19`. URL: `https://doi.org/10.1109/3PGCIC.2011.19` (cited on pages 14, 30).

[105] Stelios Sotiriadis, Nik Bessis, Pierre Kuonen, and Nick Antonopoulos. “The Inter-cloud Meta-scheduling (ICMS) Framework”. In: *27th IEEE International Conference on Advanced Information Networking and Applications, AINA 2013, Barcelona, Spain, March 25-28, 2013.* 2013, pages 64–73. DOI: `10.1109/AINA.2013.122`. URL: `https://doi.org/10.1109/AINA.2013.122` (cited on pages 28, 30, 31).

[106] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms.* Prentice-Hall, 2007 (cited on page 13).

[107] Cheikhou Thiam, Georges Da Costa, and Jean-Marc Pierson. “Cooperative Scheduling Anti-load Balancing Algorithm for Cloud: CSAAC”. In: *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 1.* 2013, pages 433–438. DOI: `10.1109/CloudCom.2013.63`. URL: `https://doi.org/10.1109/CloudCom.2013.63` (cited on pages 26, 28, 29).

[108] *Use Cases and Functional Requirements for Inter-Cloud Computing.* Global Inter-Cloud Technology Forum, GICTF White Paper. Available at: `http://www.ttc.or.jp/files/8614/1214/5480/GICTF_Whitepaper_20100809.pdf`. 2010 (cited on page 14).

[109] Ekhiotz Jon Vergara, Simin Nadjm-Tehrani, and Mikael Asplund. “Sharing the Cost of Lunch: Energy Apportionment Policies”. In: *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet 2015, Cancun, Mexico, November 2-6, 2015.*

2015, pages 91–97. DOI: 10.1145/2815317.2815338. URL: http://doi.acm.org/10.1145/2815317.2815338 (cited on page 26).

[110] Jordi Vilaplana, Francesc Solsona, Ivan Teixido, Jordi Mateo, Francesc Abella, and Josep Rius Torrento. “A queuing theory model for cloud computing”. In: *The Journal of Supercomputing* 69.1 (2014), pages 492–507. DOI: 10.1007/s11227-014-1177-y. URL: https://doi.org/10.1007/s11227-014-1177-y (cited on page 36).

[111] Leping Wang and Ying Lu. “An Efficient Threshold-Based Power Management Mechanism for Heterogeneous Soft Real-Time Clusters”. In: *IEEE Transactions on Industrial Informatics* 6.3 (2010), pages 352–364. ISSN: 1551-3203. DOI: 10.1109/TII.2010.2047950 (cited on page 19).

[112] Xiaoli Wang and Yuping Wang. “Energy-efficient Multi-task Scheduling Based on MapReduce for Cloud Computing”. In: *Seventh International Conference on Computational Intelligence and Security, CIS 2011, Sanya, Hainan, China, December 3-4, 2011*. 2011, pages 57–62 (cited on page 2).

[113] Mark Weiser, Brent B. Welch, Alan J. Demers, and Scott Shenker. “Scheduling for Reduced CPU Energy”. In: *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI), Monterey, California, USA, November 14-17, 1994*. 1994, pages 13–23. URL: http://dl.acm.org/citation.cfm?id=1267640 (cited on page 25).

[114] Josh Whitney and Pierre Delforge. “Issue Paper: Data Center Efficiency Assessment Scaling up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers”. In: *Natural Resources Defense Council (NRDC)* (2014) (cited on page 2).

[115] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, and Rajkumar Buyya. "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications". In: *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April 2010*. 2010, pages 446–452. URL: `https://doi.org/10.1109/AINA.2010.32` (cited on page 56).

[116] Chia-Ming Wu, Ruay-Shiung Chang, and Hsin-Yu Chan. "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters". In: *Future Generation Comp. Syst.* 37 (2014), pages 141–147 (cited on pages 26, 28, 32, 37).

[117] Alice Yalaoui, Hicham Chehade, Farouk Yalaoui, and Lionel Amodeo. *Optimization of logistics*. John Wiley & Sons, 2012 (cited on page 21).

[118] Frances F. Yao, Alan J. Demers, and Scott Shenker. "A Scheduling Model for Reduced CPU Energy". In: *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*. 1995, pages 374–382. DOI: `10.1109/SFCS.1995.492493`. URL: `http://dx.doi.org/10.1109/SFCS.1995.492493` (cited on pages 9, 25, 37, 121, 123, 125, 141).