electronics

MDPI

Article

A Multiformalism-Based Model for Performance Evaluation of Green Data Centres

Enrico Barbierato [1,*,†], Daniele Manini [2,†] and Marco Gribaudo [3,†]

[1] Dipartimento di Matematica e Fisica, Università Cattolica del Sacro Cuore, Via della Garzetta 48, 25133 Brescia, Italy
[2] Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10124 Torino, Italy
[3] Dipartimento di Elettronica, Informatica e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy
* Correspondence: enrico.barbierato@unicatt.it
† These authors contributed equally to this work.

Abstract: Although the coexistence of ARM and INTEL technologies in green data centres is technically feasible, significant challenges exist that must be addressed. These challenges stem from the differences in instruction sets and power consumption between the two processor architectures. While ARM processors are known for their energy efficiency, INTEL processors tend to consume more power. Consequently, evaluating the performance of hybrid architectures can be a complex task. The contributions of this article consist of (i) a multiformalism-based model of a data centre, providing a natural and convenient approach to the specification process and performance analysis of a realistic scenario and (ii) a review of the performance indices, including the choice of one architecture over another, power consumption, the response time, and request loss, according to different policies. As a result, the model aims to address issues such as system underutilization and the need to estimate the optimal workload balance, thereby providing an effective solution for evaluating the performance of hybrid hardware architectures.

Keywords: hybrid computing; ARM; INTEL; multiformalism

Citation: Barbierato, E.; Manini, D.; Gribaudo, M. A Multiformalism-Based Model for Performance Evaluation of Green Data Centres. *Electronics* **2023**, *12*, 2169. https://doi.org/10.3390/electronics12102169

Academic Editors: Jesús Ángel Román Gallego, María-Luisa Pérez-Delgado, Alfonso Jose Lopez Rivero, María Concepción Vega Hernández and Daniel Hernández De la Iglesia

Received: 31 March 2023
Revised: 6 May 2023
Accepted: 7 May 2023
Published: 10 May 2023

1. Introduction

A data centre is a centralised location where computer systems (and associated components), regarded as telecommunications and storage systems, are housed. Apart from being a hub for management and storage, a data centre serves a more general purpose, consisting of the dissemination of data and information for organisations, businesses, and individuals. Furthermore, data centres can host websites, run software applications, and offer cloud computing services. Security, scalability, and redundancy are the criteria behind the hardware architectures. In this sense, data centres typically consist of a large number of servers and other computing equipment that are organised into racks and housed in a large room or building. A remarkable drawback is represented by the significant amount of energy consumed due to the electricity required for powering and cooling the equipment and, secondly, the water needed for cooling. Inevitably, this can lead to serious environmental impacts, including greenhouse gas emissions and water usage (Table 1 summarises a number of other problems associated with data centres).

Because data centres are critical to the functioning of a company, they also present a number of challenges derived from the technology adopted to build the key parts of a data centre, as per Table 2.

Table 1. Issues associated with data centres.

Issue	Risk
Cooling and energy consumption	High electricity bills and environmental concerns
Security	Unauthorised access
Maintenance and repair	Time-consuming and possibly expensive process
Capacity and scalability	Significant investments in new hardware and infrastructure
Disaster recovery	Profit loss without a proper recovery plan and robust architecture

Table 2. Components of a data centre.

Component	Description
Servers	The physical computer systems that enforce data processing and storage.
Networking equipment	includes switches, routers, firewalls, and other components connecting servers to each other and to the external world.
Storage devices	Used to store vast amounts of data, such as hard disk drives (HDDs), solid-state drives (SSDs), and possibly tape drives.
Power and cooling systems	Data centres require an impressive amount of power to work. As a result, they generate a significant amount of heat; therefore, cooling systems are necessary to prevent overheating.
Backup and redundancy systems	To ensure that data are available on a 24/7 basis, data centres often have backup power supplies, backup networking equipment, and redundant storage devices and servers.
Physical security	Stored data contain sensitive and valuable information, so physical security measures, such as biometric access controls, should be used.
Management software	Data centres require software to manage and monitor the networking equipment, servers, and storage devices.

The backbone of a data centre is its network infrastructure, which includes high-speed networking equipment such as switches, routers, and fibre optic cables. The network backbone connects all servers, storage devices, and other equipment within the data centre and provides a high-bandwidth, low-latency communication pathway between them. On the other hand, servers are an essential component of any data centre. They play a crucial role in the storage, processing, and dissemination of data within the data centre and beyond. The importance of server technology in a data centre can be articulated according to the following points:

- Storage: Servers are responsible for storing vast amounts of data, from mission-critical business data to personal photos and videos. Without servers, data centres would not be able to store and manage the enormous amount of data that is generated every day.
- Processing: Servers also provide the processing power required to perform complex operations on the data stored in the data centre. This includes running applications, analysing data, and performing calculations.

- Scalability: Servers can be easily scaled up or down depending on the needs of the data centre. As the amount of data grows, additional servers can be added to the data centre to handle the increased workload.
- Reliability: Servers are designed to be highly reliable and available. They are built with redundant components and are monitored 24/7 to ensure they are functioning correctly.
- Security: Servers are also responsible for ensuring the security of the data stored in the data centre. They provide features like encryption, access controls, and firewalls to protect against unauthorised access and cyber threats.

There are various types of processors used in data centres. They offer high core counts, large memory capacities, and advanced security features. They are known for their energy efficiency and performance-per-dollar value. IBM Power processors are designed for high-performance computing and are commonly used in supercomputers and large data centres. They offer a high processing power, scalability, and reliability. Other processors used in data centres include NVIDIA GPUs for machine learning and AI workloads and custom processors designed by companies such as Google and Amazon for their specific data centre needs.

The mix of processors used in a data centre can vary depending on the specific needs and requirements of the data centre. However, in general, data centres often use a combination of traditional x86 processors and specialised processors, such as GPUs and ASICs. X86 processors, which are made by companies such as INTEL and AMD, are commonly used in data centres because they are general-purpose processors that can handle a wide variety of tasks. These processors are good at handling tasks that require a high single-thread performance, such as running databases or virtual machines. Specialised processors, such as GPUs and ASICs, are often used in data centres to handle tasks that require high parallelism, such as machine learning or data analytics. GPUs, which are made by companies such as NVIDIA, are particularly well-suited for these tasks, because they have a large number of cores that can perform calculations in parallel. ASICs, or application-specific integrated circuits, are designed for specific tasks and are often used in data centres for tasks such as networking, storage, and security. INTEL has been in the industry for over five decades, producing some of the most advanced and powerful processors for personal computers, servers, and other electronic devices. INTEL proposed the development of the x86 processor architecture, which is widely deployed in desktops, laptops, and servers. This technology is known for different reasons, such as its high performance, reliability, and compatibility with various software and hardware systems. A wide range of operating systems, including Microsoft Windows and Linux, are also compatible with INTEL chips. As a drawback, INTEL technology is also known for its relatively high power consumption, which can result in higher heat generation, a lower battery life, and increased energy costs.

In contrast to INTEL, ARM technology is a type of processor architecture oriented toward mobile devices, such as smartphones and tablets. Notably, ARM processors have been designed to be low-power and energy-efficient, offering a longer battery life. Differently from INTEL's x86 architecture, which is a complex instruction set computing (CISC) architecture, ARM processors exploit a reduced instruction set computing (RISC) architecture. Due to its strength, ARM technology has been widely adopted in the mobile device industry due to its low power consumption, smaller size, and lower cost. Furthermore, this technology has several advantages, ranging from its low power consumption to its energy efficiency and cost-effectiveness. These aspects make ARM technology perfectly suitable for use in mobile devices and IoT devices (as battery life and energy efficiency are of paramount importance). However, traditionally, ARM processors have been less powerful than INTEL processors, a factor that can limit their deployment in high-performance computing applications.

The integration of ARM and INTEL technologies provides several advantages, starting from balancing performance and energy efficiency in devices. However, this integration

raises a cost, which can be described in terms of the differences in instruction sets, power consumption, system architecture and finally in compatibility issues between software designed for one architecture and the other. An additional cost to be considered is related to the fact that manufacturers need to design custom systems to optimise performance and power consumption. Several devices have acquired ARM/INTEL hybrid technology to balance performance and energy efficiency. Examples of this integration consist of some laptops deploying ARM processors for low-power tasks (typically running operating systems or handling background processes), while favouring INTEL processors for high demanding tasks such as running applications or playing video games. in this way, manufacturers succeed in balancing power consumption and performance while still providing an excellent user experience. In addition, some data centres have integrated ARM and INTEL processors optimising performance and energy efficiency, i.e., using ARM processors for low-power tasks and INTEL processors for high-performance tasks. In the near future, it is likely that ARM/INTEL hybrid technologies will prevail in many devices, although manufacturers will optimise performance, power consumption, and compatibility.

A model is a simplified representation of a complex system that captures its essential features and behaviours. In this sense, modelling a system is a useful approach to evaluate its performance, because it allows for a comprehensive and accurate representation of the system's behaviour. By constructing a model of a system, it is possible to analyse its performance, identify potential issues, and optimise its behaviour. The process of modelling a system typically involves creating a mathematical or computational representation of the system that captures its key characteristics, such as its inputs, outputs, internal states, and interactions with the environment. This model can then be used to simulate the behaviour of the system under different conditions, such as different levels of demand, changes in operating parameters, the occurrence of faults or failures, and recovery to a normal state. One key advantage of modelling a system is that it allows for the identification of potential bottlenecks or areas of inefficiency in the system. By simulating the system under different conditions, it is possible to identify areas where the system may be underutilised or overloaded and to optimise the system's behaviour to achieve a better performance. Additionally, modelling a system can help to identify potential failures or faults in the system and to evaluate the system's reliability and safety. Finally, modelling a system is a powerful tool for evaluating its performance, because it allows for a detailed analysis of the system's behaviour and enables the optimisation of its performance. By constructing an accurate model of the system, it is possible to gain insights into its behaviour that would be difficult or impossible to obtain through experimentation or observation alone.

Multiformalism modelling is a powerful approach for analysing complex systems that integrates multiple formalisms to represent different aspects of the system. It is a natural and convenient method for the specification and analysis of performance, dependability, and security of systems. The approach allows the combination of different formalisms, such as Petri nets, process algebras, and queuing networks, to capture different system behaviours and characteristics. Multiformalism modelling has been widely used in various fields, including computer science, engineering, and management, to analyse and optimise systems' performance and reliability. One of the main advantages of multiformalism modelling is its ability to represent the system at multiple levels of abstraction. This allows a more comprehensive and accurate representation of the system, which can facilitate the analysis and optimisation of the system's behaviour. Moreover, multiformalism modelling can help to identify potential bottlenecks, faults, and vulnerabilities in the system, which can be addressed before they become critical issues. Furthermore, multiformalism modelling can be applied to a wide range of systems, including software systems, communication networks, manufacturing systems, transportation systems, and biological systems. The approach has been used to optimise the performance of large-scale data centres, to evaluate the reliability of safety-critical systems, and to analyse the behaviour of complex biological systems.

There are several commonly used metrics to benchmark the performance of a data centre. For instance, the power usage effectiveness (PUE) is a measure of how efficiently a data centre uses energy. It is calculated by dividing the total energy used by the data centre by the energy used by the IT equipment. The data centre infrastructure efficiency (DCIE) is a metric that indicates the percentage of energy used by the IT equipment compared to the total energy used by the data centre. It is calculated by dividing the energy used by the IT equipment by the total energy used by the data centre. Statistical measures are often used to benchmark a data centre, for example, the mean time between failures (MTBF, a measure of how often hardware in the data centre fails, that is calculated by dividing the total operating time by the number of failures) and the mean time to repair (MTTR, a measure of how long it takes to repair hardware in the data centre after a failure, that is calculated by dividing the total downtime by the number of failures). However, some measures can provide equally interesting performance indices, such as the availability, which measures the percentage of time that a data centre is operational and able to provide access to IT services, and server utilization, a measure of how much of a server's processing power is being used. Higher utilization rates generally indicate better efficiency. From an engineering perspective, it is important to consider the network latency, which measures the delay in the transmission of data between two points on a network. Lower latency is generally better for applications that require real-time responsiveness. However, in a hybrid server architecture, where the correct solution of which machine has to be chosen to fulfil a specific task, different benchmarks should be used (see Section 3.1 for a discussion). The significance of this paper lies in its two-fold contribution: firstly, the proposal of a multiformalism-based model for a data centre, which presents a user-friendly and efficient method for specifying and analysing performance in a practical setting and, secondly, a comprehensive assessment of performance metrics, encompassing factors such as architecture selection, power utilisation, response time, and request loss, in accordance with various policies.

This work is organised as follows: Section 2 presents an overview of scientific literature with regard to hybrid architectures deployed in data centres and multiformalism modelling. Section 3.1 introduces a case study. Section 4 reviews a multiformalism model, while Section 5 analyses the results from the experiments. Finally, Section 6 draws the conclusion and the lines of research for future work.

2. Related Work

The review of the scientific literature is presented as follows. Section 2.1 reviews the technology behind heterogeneous computing with regard to the ARM and INTEL components. Section 2.2 discusses notable approaches to the simulation of data centres, and finally, Section 2.3 reviews literature about multiformalism. It should be noticed that the references cited in Section 2.2 do not exploit multiformalism techniques.

2.1. Hybrid ARM/INTEL Architectures

Heterogeneous computing (HC) has different applications in various domains, such as big data analytics, machine learning, and scientific computing. By using both ARM and INTEL architectures, HC can provide a flexible and scalable solution to the increasing demands of modern computing applications.

HC architectures can be evaluated by benchmarks in order to provide interesting insights and reveal design weaknesses. For example, Che et al. discussed Rodinia [1], a benchmark suite for HC oriented toward parallel communication patterns, synchronisation techniques, and power consumption analysis. Danalis et al. presented Scalable Heterogeneous Computing (SHOC) [2], a benchmark suite deployed in systems including graphics processing units (GPUs) and multicore processors. Performance is measured in terms of specific system features, such as communication between devices. Aroca et al. [3] proposed a testing setup to verify the power and performance of an architecture based on several servers, such as web and database servers, and X86 and ARM nodes by using a Linpack benchmark on floating point operations. Using green IT as the main driver,

the authors proved that is possible to diminish data centre power usage by putting x86 servers to sleep and then restoring them with Wake On LAN when the computing demand increases. HC pursues different approaches, for example, CPUs and GPUs can be paired to be deployed in a vast range of applications in order to improve both performance and energy consumption [4]. Brodtkorb et al. conducted an overview of the state-of-the-art HC technology [5], focussing on three typical architectures (specifically, the Cell Broadband Engine Architecture, GPUs, and field programmable gate arrays or FPGAs). The authors criticised the use of symmetric multiprocessing as a long term approach, instead supporting architectures based on accelerator cores coupled with a set of traditional CPU cores. An overview of the challenges and opportunities related to HC using ARM and INTEL architectures can be found in [6]. The authors discuss the challenges of programming and optimising heterogeneous systems, including the need for a proper algorithm design, profiling of the embedded parallelism, parallelism detection, analytical benchmarking, and hardware selection, i.e., identifying the most suitable heterogeneous machines to host a set of applications whose execution is bound by specific constraints. Padoin et al. [7] explored the performance and energy consumption trade-offs in scientific computing using two different architectures: ARM big.LITTLE and INTEL Sandy Bridge. In particular, the article compares the big.LITTLE architecture, a combination of high-performance and low-power cores, with the Sandy Bridge architecture, which uses a homogeneous design with high-performance cores. The authors performed a series of experiments to compare the performance and energy consumption of the two architectures when running scientific computing applications. The results were mixed, as the big.LITTLE architecture achieved a better energy efficiency than the Sandy Bridge architecture when running certain types of scientific computing applications. However, the performance of the big.LITTLE architecture was generally worse than that of the Sandy Bridge architecture, especially for applications that require a lot of computational power.

A more comprehensive performance evaluation was performed by Jarus et al. [8], where a series of experiments was conducted using a cluster of HPC nodes equipped with the three different processor architectures. The authors analysed the performance of the processors using various benchmarks and workloads, including the LINPACK benchmark, the HPL benchmark, and molecular dynamics simulations. The outcome of the experiments shows that the INTEL processors outperformed the AMD and ARM processors in terms of their raw performances. On the other hand, the ARM processors are more energy-efficient than the INTEL and AMD processors, achieving a better performance per watt of energy consumed.

ARMINTEL [9] is a novel microprocessor architecture that enables INTEL-based applications to run on ARM-based microprocessors. ARMINTEL is described as a heterogeneous microprocessor architecture that combines the advantages of both INTEL and ARM architectures. It consists of two types of cores: INTEL x86 cores and ARM cores. The INTEL x86 cores are responsible for executing INTEL-based applications, while the ARM cores handle other tasks. It is argued that ARMINTEL has several advantages over existing solutions. Firstly, it enables INTEL-based applications to run natively on ARM-based microprocessors, eliminating the need for emulation or virtualisation. Secondly, it provides a flexible platform that can dynamically allocate resources between x86 and ARM cores based on workload requirements. Finally, it allows for efficient power management by using ARM cores for low-power tasks and x86 cores for high-performance tasks.

2.2. Modelling Data Centres

In [10], the authors provide an overview of the aspects related to the modelling of data centres' energy consumption. In detail, the authors mapped the energy consumption of a data centre and its components to a power model, which was further described in terms of accuracy, speed, generality and portability, inexpensiveness and simplicity. Besides discussing classic power models (such as additive and utilisation models), the authors

discussed more specific proposals. Physical issues, such as airflow, heat transfer, and cooling systems, are discussed in [11,12].

With concern to the simulation and experimentation of Cloud computing infrastructures, CloudSim [13] provides two main ways to support the modelling and simulation of large-scale Cloud computing infrastructures, which include data centres within a single physical computing node. Additionally, it offers a self-contained platform that models data centres, service brokers, scheduling, and allocation policies. CloudSim boasts unique features, such as a virtualisation engine that facilitates the creation and management of multiple independen, and co-hosted virtualised services on a data centre node. Additionally, CloudSim can switch between space-shared and time-shared allocation of processing cores to virtualised services.

Luo et al. [14] presented a simulation model called CloudSim-Power, which is an extension of the CloudSim [13] simulation framework. The CloudSim-Power model simulates the behaviour of individual components in a cloud data centre, including servers, switches, storage devices, and cooling systems, and models their power consumption in real-world data centres. The model includes a scheduling algorithm that allocates resources to tasks in a way that minimises the total power consumption of the data centre while meeting the performance requirements of the tasks. The authors also show that the scheduling algorithm included in the CloudSim-Power model can effectively allocate resources to tasks in a way that minimises the total power consumption of the data centre while meeting the performance requirements of the tasks.

Prevost et al. [15] introduced a model based on an autoregressive linear prediction and neural network prediction (trained by using the backpropagation algorithm) to forecast the load demand profiles, successfully predicting the workload and estimating the number of needed resources to minimise the power consumption.

Meisner et al. [16] introduced a new methodology called the Statistical Queuing Simulation (SQS), a stochastic discrete-time simulation of generalised queueing models, characterised by arrival times and service distribution. The SQS was tested against three workloads (web mail, interactive login, and an Apache-based web server), and a case study of a data centre power provisioning technique for a 1000 server cluster was conducted. A stochastic model to size a data centre in order to minimise energy consumption is discussed in [17], where the authors claim that, by solving the model, the trade-off between the energy consumption and the model performance cannot be avoided. The trade-off was optimised by implementing a Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, while experiments were performed by adopting the Stochastic Right-Sizing Model (SRM), showing that if the performance level is coherent with what is expected from the scientific literature, the energy saving obtained is rather important.

2.3. Multiformalism Modelling

Early multiformalism modeling can be found in [18], where Generalised Stochastic Petri Nets (GSPN) and Queueing Networks (QNs) owning a product-form are mapped into the corresponding Continuous Time Markov Chain (CTMC) representation and numerically solved. The model built in this way is used to simulate classical problems in computer science, such as simultaneous resource possession, concurrent execution and synchronisation tasks. Bause et al. [19] presented a novel formalism called QPN, composed of QNs and PNs, to improve the flexibility of these formalisms by introducing timed transitions and timeless queues. The author reviewed the multiformalism in light of fundamental qualitative analysis, i.e., the boundedness, liveness, and the existence of home states. Model transformation refers to the process of converting a model from one representation to another. This process can involve changing the structure of the model, the type of modelling language used, or the level of abstraction. Model transformation is a key step in many modelling tasks, such as model validation, simulation, optimisation, and code generation. ATOM^3 [20] is a tool that supports model transformation through a graphical user interface (GUI) and a set of transformation rules. With ATOM^3, users can define transformation rules that specify

how models in one language can be transformed into models in another language. These rules are represented graphically as diagrams, which can be created and edited using the GUI. Once the transformation rules have been defined, ATOM^3 can automatically apply them to convert models from one language to another. ATOM^3 provides a set of built-in transformation rules, as well as the ability to define custom rules. This makes it a powerful tool for model transformation tasks in various domains, such as software engineering, systems engineering, and business process modelling. Multiformalism has been applied in different ways and by using dedicated tools. For example, SMART (Symbolic Model checking Analyser for Reliability and Timing, [21]) includes both stochastic models and logical analysis. SMART offers a high-level formalism (Petri Nets) and two low-level formalisms (discrete-time and continuous-time Markov Chains, i.e., DTMC and CTMC) to the modeller, who is free to calculate a set of measures for each model and exchange parameters between models. SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) is a tool that is used to analyse stochastic models [22], the most notable being fault trees, the product of Queueing Networks, Markov Chains and Generalised Stochastic Petri Nets. The tool provides a set of algorithms and a specification language to build single models or combinations of models according to a hierarchy, i.e., the output of a model is regarded as the input of another model. OsMoSys [23] is a multiformalism, multisolution modelling framework oriented toward objects. OsMoSys offers both explicit and implicit multiformalism (respectively occurring when the different modelling languages are specified by the modeller or by the tool/framework) and the ability to compose models within a single formalism. Möbius () allows the user to compose multiformalism models by representing the state of a model using a state variable, which is characterised by a type.

Based on this groundbreaking work, multiformalism has been utilised in various forms with the aid of specialised tools. One example is the SMART (Symbolic Model Checking Analyser for Reliability and Timing, ref. [21]), which employs both stochastic models and logical analysis. SMART provides a high-level formalism, namely Petri Nets, and two low-level formalisms, discrete-time and continuous-time Markov Chains (DTMC and CTMC, respectively), to enable the modeller to compute a set of metrics for each model and exchange parameters between models. Another tool, SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator, ref. [22]), was designed to analyse stochastic models, including fault trees, product-form Queueing Networks, Markov Chains, and Generalised Stochastic Petri Nets. This tool offers a set of algorithms and a specification language to construct single or hierarchical combinations of models, where the output of one model serves as the input of another. OsMoSys is a modelling framework oriented toward objects that supports explicit and implicit multiformalism, where the modelling languages are specified by the user or the tool/framework. OsMoSys [23] also enables the composition of models within a single formalism. Möbius [24] enables the user to construct multiformalism models by representing the state of a model using a state variable that is characterised by a type. SIMTHESys [25] is a framework oriented toward the definition of domain-oriented modelling languages and the automatic generation of related solvers. Finally, Java Modelling Tools (JMT [26]) is a suite of applications meant to provide a comprehensive framework that includes performance evaluation and system modelling using analytical and simulation techniques. Other features include capacity planning and workload characterisation.

3. Case Study

3.1. Introduction

The ensuing case study exemplifies a situation conveyed by the elements discussed in Sections 2.1–2.3. The intention is to present a case study of a data centre that is purposefully engineered to mitigate its carbon footprint while preserving its ability to fulfil the demands of computationally intensive operations required by commercial clients. The present study is structured into distinct stages: first, a description of the data centre's requisite server

functionality is provided. Subsequently, a multiformalism model is advanced and subjected to simulation to quantify a suite of performance indices. Lastly, an analysis of the results obtained from the multiformalism simulation is presented. Coherently with an energy-saving perspective, most data centre components (such as the network) are not included in the model, which focuses on the alternative usage of AMD and INTEL architectures according to demand by evaluating the energy cost of the selected hardware architecture.

3.2. A Green Data Centre

A new data centre was constructed to provide a diverse range of services, such as video encoding and in-memory databases. In order to comply with the internal regulations of the country, the data centre must adopt a green policy that prioritises energy conservation and reduces its carbon footprint. To achieve this, a heterogeneous architecture using both ARM and INTEL processors is required to build Virtual Machines (VMs) when necessary. The utilisation of mixed hardware is essential due to the fact that certain lightweight applications may be more efficiently executed by ARM-based VMs, while more complex computational applications require the use of INTEL-based VMs. The incorporation of a renewable energy source, such as a solar panel, contributes to the enhancement of the green data centre architecture.

In the considered scenario, it is assumed that INTEL-based VMs are faster than ARM-based VMs and each job can require multiple VMs. The service time depends on the workload type and architecture. The utilisation of the stations representing the VMs is taken as a performance index, which is used, in turn, to derive the energy consumption, using classic formulas that relate server energy consumption to its utilisation. The following step takes into account the response time, which depicts how long a VM takes to function in the selected environment. The immediate transitions are used to represent the different policies that are the subject of this study:

- INTEL priority and ARM priority, regardless of the resource availability. These policies take a VM of a certain type if it is available; otherwise, a switch to the alternative policy is taken;
- Random policy: if there are multiple machines of a particular type available, one is randomly chosen with a probability of $p = 0.5$. If a resource type is not available, the policy applies to the other one).
- A policy that selects the resource with the most availability (if there are more ARM-based than INTEL-based VMs, then ARM-VMs will be chosen, and vice versa when there are more INTEL-based VMs).
- A further policy considers the added solar panel. It then selects the INTEL architecture when the renewable source is providing energy, and it reroutes the VMs to the ARM architecture when only the main source of electricity is available.
- The last policy is a mixture of three of the previous ones. When the renewal sources are available, it gives priority to INTEL VMs. However, if there are no more INTEL VMs available, it executes a VM on an ARM architecture. Conversely, when no renewable energy is available, this policy gives priority to ARM VMs, resorting to INTEL VMS only if the former ones are all in use.

4. The Multiformalism Model

Introduction

The authors built a novel model by utilising a multiformalism approach that incorporates a fusion of GSPNs and QNs to comprehensively represent the performance of distinct green data centre ARM/INTEL technology-based servers and their inter-relationships. In the most general scenario, each task is a complex application that could require deployment on multiple interconnected servers, each running on a different virtual machine.

In detail, four types of traffic were considered: INTEL-specific and ARM-specific represent tasks that need to be executed on a virtual machine with a specific hardware architecture. Two other types of jobs instead consider tasks that can be deployed on virtual

machines with different architectures: coherent tasks, and cooperative tasks. The difference between the two types occurs whenever a job requires more than a single virtual machine to be deployed: in the former case, all virtual machines belong to the same architecture. In other words, they must either be all INTEL or all ARM. In the case of cooperative tasks, each virtual machine supporting the application can independently be INTEL- or ARM-based.

The four types of workloads were modelled with four specific job classes, starting from the corresponding queuing network source nodes (blue squares with a white "S" inside), which were labelled INTEL only, ARM only, coherent architecture, and cooperative architecture, respectively.

The proposed multiformalism model inherits source, fork (circles with K-shaped arrows), join (mirrored K-shaped arrows), queue and sink (black circle inside a blue square) nodes from Queuing Networks, while it uses places and transitions from Petri Nets.

Fork and join nodes are used to model the requirements of applications consisting of more than one virtual machine. Hardware-specific traffic is immediately routed to the corresponding type of server (INTEL and ARM classes), after being forked in the required number of Virtual Machines. Traffic belonging to the coherent and cooperative classes is routed, respectively, to places P_1 and P_2, where two couples of immediate transitions (*Choose Intel*$_1$ or *Choose ARM*$_1$, and (*Choose Intel*$_1$ or *Choose ARM*$_1$) implement the considered architecture selection policies. In particular,

1. In INTEL (resp. ARM) priority transitions, *Chose Intel*$_i$ (resp. *Choose ARM*$_i$) fires if there are INTEL (resp. ARM) virtual machines available; otherwise, *Choose ARM*$_i$ (resp. *Chose Intel*$_i$) is selected.
2. In the random policy, whenever VMs of both types are available, transitions *Chose Intel*$_i$ and *Choose ARM*$_i$ fire randomly with equal probabilities. However, if one of the resources is missing, only the other will fire.
3. In the most available resource policy, only the transition belonging to the type with more VMs is enabled. If both architectures are characterised by the same number of VMs, the choice is random.
4. The fifth policy alternates between the two priority policies based on the state of the renewable source (submodel in the bottom right of Figure 1). In particular, it inhibits the selection of the ARM architecture when the renewable source is on: in this way, VMs are routed to the INTEL architecture. Conversely, when the renewable source is off, the selection of INTEL VMs is inhibited.
5. The last policy is realised by duplicating the immediate transitions that perform the choice, implementing both the INTEL and ARM priority policies. The inhibitor ARC allows the proper pair of transitions to be enabled, according to the state of the renewable source.

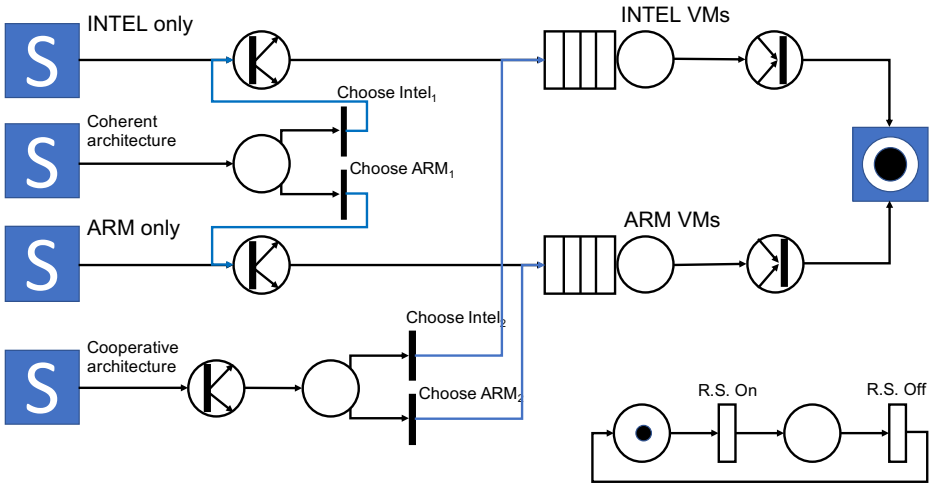

Figure 1. The multiformalism model of a hybrid processor architecture.

The main difference between the coherent and cooperative architectures is that the former checks that enough VMs of the considered type are available for all tasks required by the job, while the latter allows a separate architecture selection for each task of a job. The execution of the services is modelled by queues INTEL VMs and ARM VMs, each one denoted by an M/M/K/K queue, where K corresponds to the maximum number of VMs available for the corresponding technologies. If a task arrives when the queue has reached full capacity, it is lost.

The suite of performance indices generated by the model is depicted in Table 3.

Table 3. Performance indices suite.

Performance Index	Description
Probability of choosing one architecture over another	It models a situation when INTEL servers are not available to satisfy the demand. Consequently, ARM servers are allocated
Power consumption	The different power consumption depending on the chosen policy. For example, if INTEL servers are preferred, the energy consumption tends to be higher
Response time	The total time a request spends in the system
Loss probability	The probability of losing access to a VM

The set of parameters employed by the model is summarised in Table 4.

Table 4. Model parameters.

Parameter	Value
Arrival rate: INTEL	4 VM/h
Arrival rate: ARM	4 VM/h
Arrival rate: Coherent	between 1 and 12 VM/h
Service time: INTEL job on INTEL VM	1 h
Service time: ARM job on ARM VM	1.5 h
Service time: job (coherent) on INTEL VM	1 h
Service time: job (coherent) on ARM VM	1.5 h
Average ON time of renewable source	12 h
Average OFF time of renewable source	12 h
Baseline power consumption of all the VMs	600 W
Utilisation-dependent power consumption per INTEL VM	150 W
Utilisation-dependent power consumption per ARM VM	80 W
ARM VMs	10
INTEL VMs	10

In the model depicted in Figure 1, the model transformation technique (see Section 2.3) was applied. In particular, the possibility of having jobs composed of several tasks was not considered to be sufficiently relevant to this study. Instead, we focused on single-task jobs. This simplified the model, avoiding the introduction of fork and join nodes and making both the coherent and cooperative classes identical. The models were then solved with the assistance of the JMT (https://jmt.sourceforge.net/, accessed on 30 March 2023) tool. The results of the model transformation for the random policy are shown in Figure 2. Similar

models were generated for the other policies but are not been included to simplify the presentation.

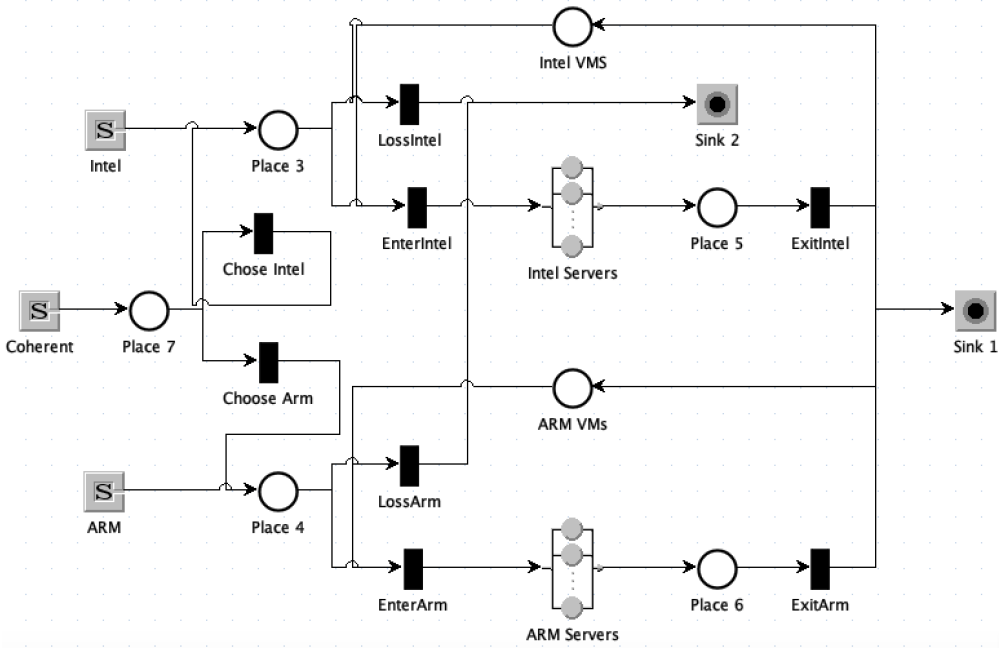

Figure 2. JMT model of the architecture shown in Figure 1 for the random policy. The other policies were implemented with similar JMT models.

In particular, M/M/K/K queues were replaced by submodels composed of an infinite server queue, a place containing the available VMs of the corresponding types, and three immediate transitions to model the entrance, the exit, and the loss events of the queue. The models were solved using a simulation, considering a 99% confidence interval, which was omitted for the sake of clarity, stopping at a maximum relative error of 3%. Finally, it has to be noted that the JMT What-If facility was performed by changing the arrival rate of the Coherent VMs, varying the requests from 4 to 12 requests per hour (the step was set to 2).

5. Discussion

Figure 3 shows the probability of choosing one architecture over another. The random policy selects the INTEL and ARM architectures equally with a probability of 0.5. The INTEL priority policy initially favours INTEL processors, but as the workload increases, the percentage of available INTEL architectures tends to decrease, because they cannot satisfy the entire workload. The use of the ARM architecture tends to increase to satisfy the demand that INTEL cannot meet. In the policy that chooses the architecture with the largest number of VMs available, INTEL is prioritised since, due to its shorter execution time, it has a larger probability of being available with respect to ARM. Both policies that consider the renewable source are symmetric, alternating equally between ARM and INTEL architectures. Due to the fact that the availability of the renewable sources is considered to be 50% of the total time, both types of VMs have a 50% probability of being chosen.

Figure 4 displays the total power consumption in watts of a data centre that was designed to accommodate 20 machines, including 10 INTEL and 10 ARM machines. When the renewable source policy is considered, it is supposed that the source is capable of supporting the entire data centre, and the power consumption considers only the time instants when only the main energy is available. It can be observed that when prioritising the use of INTEL machines, greater energy consumption occurs. As discussed in Figure 3, the architecture with the largest number of VMs available is selected and the INTEL VMs are prioritised, thus resulting in greater power consumption. Using INTEL VMs for coherent jobs only when the renewable source is active has a great impact on the power consumption.

In particular, the minimum value is achieved when INTEL VMs are used for coherent traffic only when the renewable source is available.

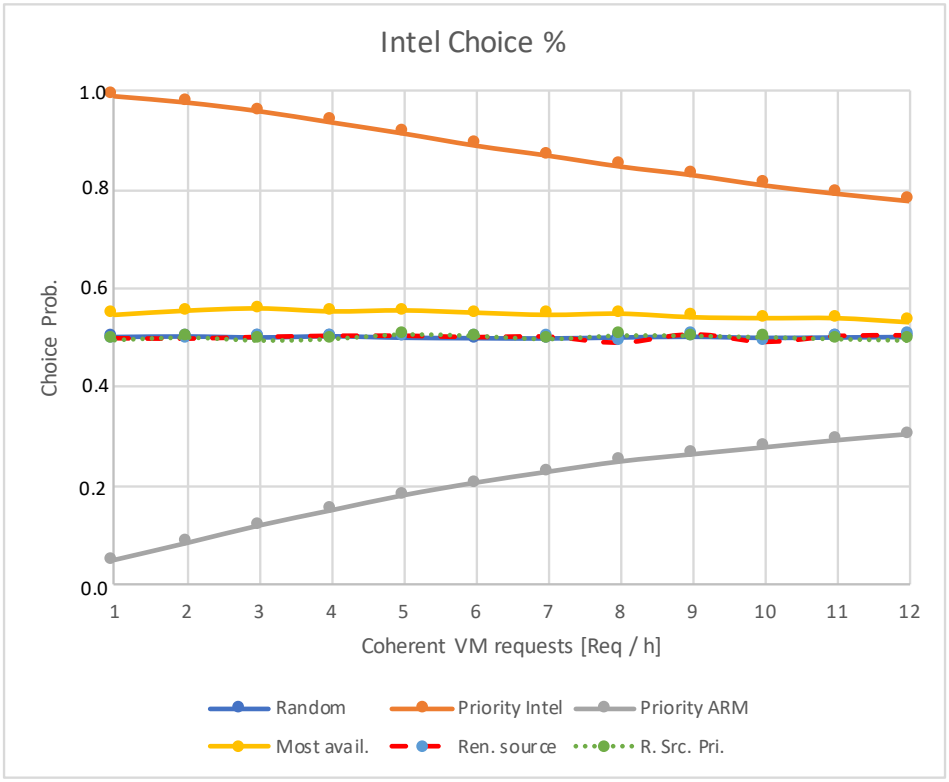

Figure 3. Probability of choosing one architecture over another.

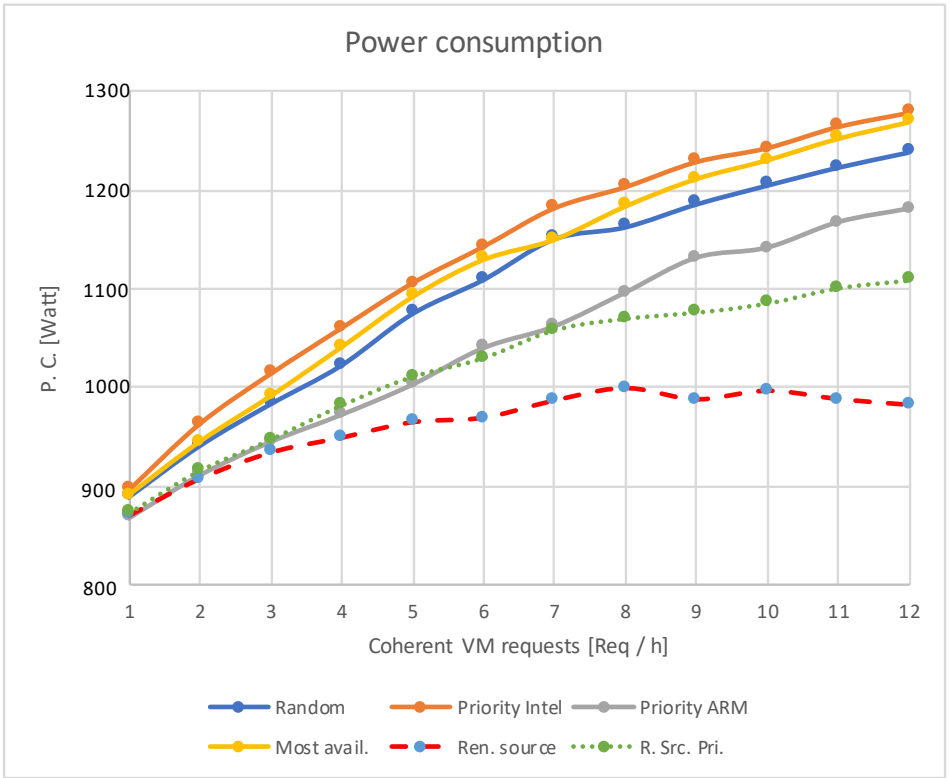

Figure 4. Power consumption.

Figure 5 shows the effect of the policies on the response time. Since the system either immediately executes a request, or drops it, no queue is actually performed and the average response time also corresponds to the average running time of a VM. As expected, it can be observed that prioritising the selection of virtual machines (VMs) based on INTEL architectures results in a decrease in the response time, as the VMs based on ARM are assumed to be 50% slower. The random policy, and the two that consider the availability of the renewable source, have the exact same response times: this is a consequence of the fact that, in all three cases, both types of resources are used equally, as shown in Figure 3. The behaviour of both the INTEL and ARM priority policies is nonmonotonic. This is caused by the fact that, as the traffic increases, more and more VMs or different technologies are used, thus making the system either slower or faster. Interestingly, it appears that as the workload increases, the response time decreases. This is a counterintuitive result, as one would expect the response time to increase with an increase in the workload. The reason for this can be found in the following graph.

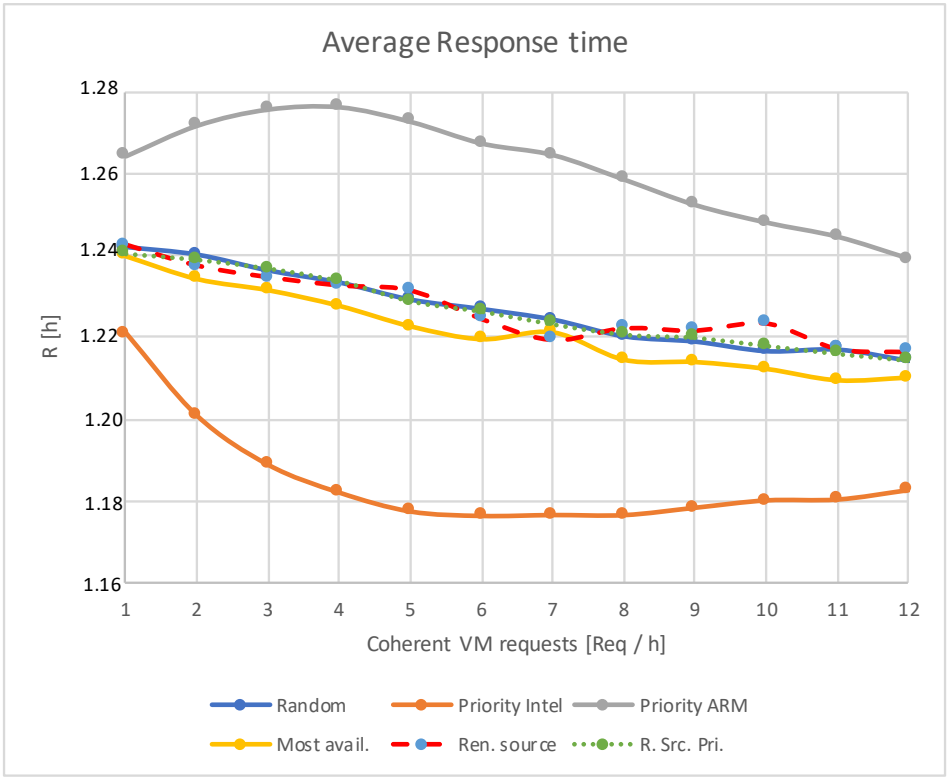

Figure 5. Response time.

Figure 6 depicts the probability of losing access to a VM and, consequently, losing the request. From the figure, it can be observed that when, considering ARM machines, the probability of loss is high due to their slower speeds compared with INTEL machines, which leads to higher saturation. Additionally, when there are more INTEL machines, as long as the workload is low, the probability of loss is lower, because these machines can execute the required tasks more quickly. Consequently, machines are released earlier and more requests can be served. As the workload increases, in terms of the probability of losing requests, the random policy seems to be the most interesting since it uniformly distributes the workload between INTEL and ARM VMs, thus reducing the probability of loss. The fact that there are still high losses is the reason why the response time decreases with an increase in the workload. The number of losses is higher, but these losses free up VMs, which leads to the few accepted requests being completed earlier. The policy that directs jobs only to the ARM VMs when the renewable source is not available is the worse in terms of availability. This is due to the longer execution time required for this technology

compared with the other. Controlling instead only the priority of choosing ARM technology according to the renewable sources produces a loss probability comparable to that of the random strategy.

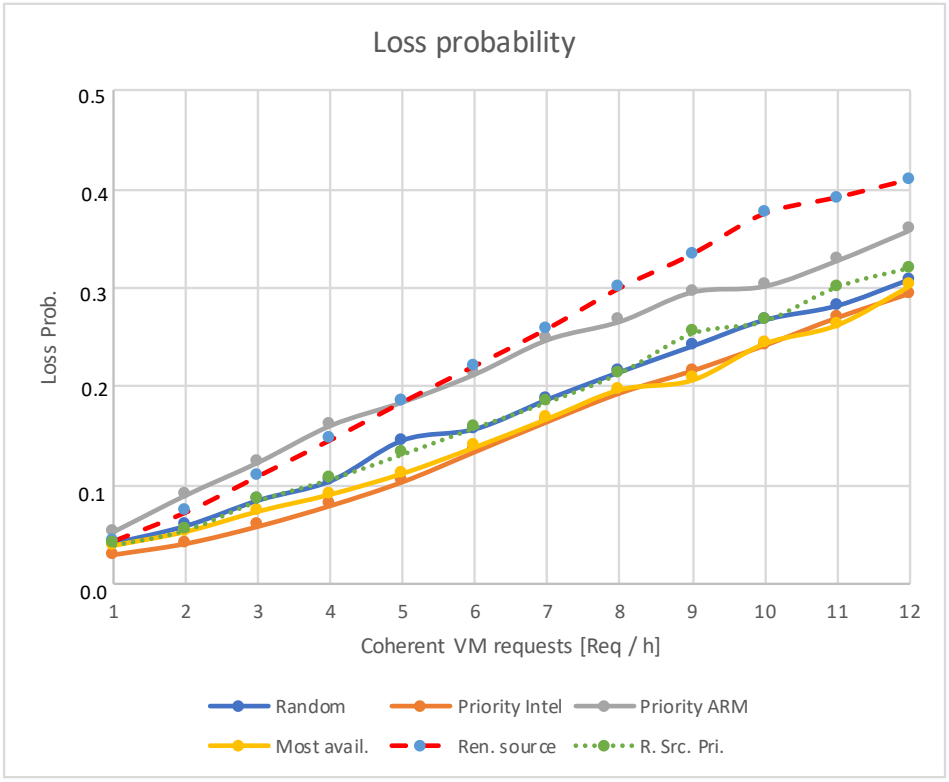

Figure 6. Loss probability.

It is worth noting that this work closely reflects the aspects related to future directions and trends undertaken by data centres [27–29], specifically (i) the use of renewable energy sources such as solar, wind, and hydroelectric power to reduce the reliance on nonrenewable energy sources and greenhouse gas emissions; (ii) data centres are designed to operate at high levels of efficiency by using advanced cooling systems, optimising server utilization, and using low-power processors. Technologies such as evaporative cooling systems and air-side economisers help to reduce water usage; (iii) ARM/INTEL hybrid data centres are becoming more popular due to their flexibility, scalability, and energy efficiency; (iv) the adoption of edge computing architectures, i.e., moving processing power closer to the source of the data, reduces the amount of data that needs to be transmitted, thus reducing energy consumption; and finally, (v) the use of sustainable materials in construction that are environmentally friendly and recyclable to reduce the carbon footprint of the data centre.

With regard to multiformalism modelling, some interesting trends [30,31] refer to multiparadigm modelling, including discrete-event, continuous-time, or agent-based modelling, to describe different aspects of a system or process and (ii) model validation and verification, which involves ensuring that models are accurate and reliable by comparing them to real-world data, running simulations, and performing sensitivity analyses.

6. Conclusions and Future Work

This paper presents a multiformalism model to simulate a green data centre based on an INTEL/ARM architecture, reproducing scenarios where the availability of VMs based on a specific processor varies according to different scenarios.

To summarise, three different goals can be achieved by choosing the most appropriate policy: reducing the power consumption (increasing the green mission of the data centre), reducing the response time (increasing the performance), and reducing the drop probability

of a VM (increase the profit of the data-centre). Giving priority to the "fast" resource can be used to obtain either the best performance or the lowest drop probability. The two policies check the availability of the renewable source and produce, as expected, the lowest power consumption. If losing traffic is not an issue, then avoiding using INTEL VMs when a renewable source is available produces the best results. If, however, losing jobs is an issue, then giving priority to ARM VMs when no renewable source is available gives the best trade-off, since both policies are characterised by similar response times. It is also interesting to note that giving priority to ARM VMs is never a good choice, since they do not provide an energy consumption reduction sufficient to motivate the reduction in both performance and availability. Choosing the most available technology is also not worth the energy consumption increase, since it only marginally increases the performance and availability compared with the other policies. Finally, when a simple solution is required, the random one seems to provide a good trade-off between performance, energy consumption, and loss probability. In future work, we aim to study more complex models, for example, adding to the model fork/join constructs and different processors.

Author Contributions: All the authors contributed equally to the manuscript in terms of conceptualization, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TLA	Three letter acronym
LD	Linear dichroism

References

1. Che, S.; Boyer, M.; Meng, J.; Tarjan, D.; Sheaffer, J.W.; Lee, S.H.; Skadron, K. Rodinia: A benchmark suite for heterogeneous computing. In Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, USA, 4–6 October 2009; pp. 44–54.
2. Danalis, A.; Marin, G.; McCurdy, C.; Meredith, J.S.; Roth, P.C.; Spafford, K.; Tipparaju, V.; Vetter, J.S. The scalable heterogeneous computing (SHOC) benchmark suite. In Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, Pittsburgh, PA, USA, 14 March 2010; pp. 63–74.
3. Aroca, R.V.; Gonçalves, L.M.G. Towards green data centers: A comparison of x86 and ARM architectures power efficiency. *J. Parallel Distrib. Comput.* **2012**, *72*, 1770–1780. [CrossRef]
4. Mittal, S.; Vetter, J.S. A survey of CPU-GPU heterogeneous computing techniques. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–35. [CrossRef]
5. Brodtkorb, A.R.; Dyken, C.; Hagen, T.R.; Hjelmervik, J.M.; Storaasli, O.O. State-of-the-art in heterogeneous computing. *Sci. Program.* **2010**, *18*, 1–33. [CrossRef]
6. Khokhar, A.A.; Prasanna, V.K.; Shaaban, M.E.; Wang, C.L. Heterogeneous computing: Challenges and opportunities. *Computer* **1993**, *26*, 18–27. [CrossRef]
7. Padoin, E.L.; Pilla, L.L.; Castro, M.; Boito, F.Z.; Alexandre Navaux, P.O.; Méhaut, J.F. Performance/energy trade-off in scientific computing: The case of ARM big.LITTLE and Intel Sandy Bridge. *IET Comput. Digit. Tech.* **2015**, *9*, 27–35. [CrossRef]
8. Jarus, M.; Varrette, S.; Oleksiak, A.; Bouvry, P. Performance evaluation and energy efficiency of high-density HPC platforms based on Intel, AMD and ARM processors. In Proceedings of the Energy Efficiency in Large Scale Distributed Systems: COST IC0804 European Conference, EE-LSDS 2013, Vienna, Austria, 22–24 April 2013; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany, 2013; pp. 182–200.
9. Madni, S.; Shafiq, M.; Rashid, H.U. ARMINTEL: A heterogeneous microprocessor architecture enabling intel applications on ARM. In Proceedings of the 2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 14–18 January 2020; pp. 370–377.

10. Dayarathna, M.; Wen, Y.; Fan, R. Data center energy consumption modeling: A survey. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 732–794. [CrossRef]
11. Rambo, J.; Joshi, Y. Modeling of data center airflow and heat transfer: State of the art and future trends. *Distrib. Parallel Databases* **2007**, *21*, 193–225. [CrossRef]
12. Zhang, Q.; Meng, Z.; Hong, X.; Zhan, Y.; Liu, J.; Dong, J.; Bai, T.; Niu, J.; Deen, M.J. A survey on data center cooling systems: Technology, power consumption modeling and control strategy optimization. *J. Syst. Archit.* **2021**, *119*, 102253. [CrossRef]
13. Calheiros, R.N.; Ranjan, R.; Rose, C.A.F.D.; Buyya, R. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. *arXiv* **2009**, arXiv:0903.2525.
14. Luo, L.; Wu, W.; Tsai, W.T.; Di, D.; Zhang, F. Simulation of power consumption of cloud data centers. *Simul. Model. Pract. Theory* **2013**, *39*, 152–171. [CrossRef]
15. Prevost, J.J.; Nagothu, K.; Kelley, B.; Jamshidi, M. Prediction of cloud data center networks loads using stochastic and neural models. In Proceedings of the 2011 6th International Conference on System of Systems Engineering, Albuquerque, NM, USA, 27–30 June 2011; pp. 276–281.
16. Meisner, D.; Wenisch, T.F. Stochastic Queuing Simulation for Data Center Workloads. In Proceedings of the Exascale Evaluation and Research Techniques Workshop, 2010; Volume 16, p. 37. Available online: https://web.eecs.umich.edu/~twenisch/papers/exert10.pdf (accessed on 5 May 2023).
17. Shen, D.; Luo, J.; Dong, F.; Fei, X.; Wang, W.; Jin, G.; Li, W. Stochastic modeling of dynamic right-sizing for energy-efficiency in cloud data centers. *Future Gener. Comput. Syst.* **2015**, *48*, 82–95. [CrossRef]
18. Balbo, G.; Bruell, S.C.; Ghanta, S. Combining queueing networks and generalized stochastic Petri nets for the solution of complex models of system behavior. *IEEE Trans. Comput.* **1988**, *37*, 1251–1268. [CrossRef]
19. Bause, F. Queueing Petri Nets-A formalism for the combined qualitative and quantitative analysis of systems. In Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse, France, 19–22 October 1993; pp. 14–23.
20. Lara, J.d.; Vangheluwe, H. AToM 3: A Tool for Multi-formalism and Meta-modelling. In Proceedings of the Fundamental Approaches to Software Engineering: 5th International Conference, FASE 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002, Grenoble, France, 8–12 April 2002; Proceedings 5; Springer: Berlin/Heidelberg, Germany, 2002; pp. 174–188.
21. Ciardo, G.; Jones, R.L., III; Miner, A.S.; Siminiceanu, R.I. Logic and stochastic modeling with SMART. *Perform. Eval.* **2006**, *63*, 578–608. [CrossRef]
22. Trivedi, K.S. SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator. In Proceedings of the 2002 International Conference on Dependable Systems and Networks, Washington, DC, USA, 23–26 June 2002; IEEE Computer Society: Washington, DC, USA, 2002; p. 544.
23. Franceschinis, F.; Gribaudo, M.; Iacono, M.; Mazzocca, N.; Vittorini, V. Towards an Object Based Multi-Formalism Multi-Solution Modeling Approach. In Proceedings of the Second International Workshop on Modelling of Objects, Components, and Agents (MOCA'02), Aarhus, Denmark, 26–27 August 2002; Technical Report DAIMI PB-561; Moldt, D., Ed.; 2002; pp. 47–66.
24. Clark, G.; Courtney, T.; Daly, D.; Deavours, D.; Derisavi, S.; Doyle, J.; Sanders, W.; Webster, P. The Mobius modeling tool. In Proceedings of the 9th International Workshop on Petri Nets and Performance Models, Aachen, Germany, 11–14 September 2001; pp. 241–250. [CrossRef]
25. Barbierato, E.; Bobbio, A.; Gribaudo, M.; Iacono, M. Multiformalism to support software rejuvenation modeling. In Proceedings of the 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops, Dallas, TX, USA, 27–30 November 2012; pp. 271–276.
26. Bertoli, M.; Casale, G.; Serazzi, G. JMT: Performance engineering tools for system modeling. *ACM SIGMETRICS Perform. Eval. Rev.* **2009**, *36*, 10–15. [CrossRef]
27. Li, Z.; Kandlikar, S.G. Current status and future trends in data-center cooling technologies. *Heat Transf. Eng.* **2015**, *36*, 523–538. [CrossRef]
28. Bilal, K.; Khalid, O.; Erbad, A.; Khan, S.U. Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. *Comput. Netw.* **2018**, *130*, 94–120. [CrossRef]
29. SG Andrae, A. New perspectives on internet electricity use in 2030. *Eng. Appl. Sci. Lett.* **2020**, *3*, 19–31.
30. Heath, J.P.; Harding, J.H.; Sinclair, D.C.; Dean, J.S. The analysis of impedance spectra for core–shell microstructures: Why a multiformalism approach is essential. *Adv. Funct. Mater.* **2019**, *29*, 1904036. [CrossRef]
31. Khan, S.; Volk, M.; Katoen, J.P.; Braibant, A.; Bouissou, M. Model checking the multi-formalism language Figaro. In Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Taipei, Taiwan, 21–24 June 2021; pp. 463–470.