UNIVERSITEIT VAN AMSTERDAM

'ODAbler': Design and Evaluation of an Operational Data Analytics Framework for Energy-efficient management of Workloads in a Data Centre Simulator OpenDC

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Computer Science

Department of Computer Science
Faculty of Science
Vrije Universiteit Amsterdam & Universiteit van Amsterdam

Research Supervisor:
Prof. dr. ir. Alexandru Iosup

Proponent:
Shekhar Suman

Daily Supervisor:
Xiaoyu Chu (PhD fellow)

Second Reader:
Prof. dr. Tiziano De Matteis

Date: October 31, 2023

Place: Amsterdam,
The Netherlands

The Road Not Taken

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

ROBERT FROST (1874–1963)

Abstract

In an era marked by the growing urgency of addressing climate change and advancing green initiatives, the role of data centres in sustainable technology practices cannot be overstated. Data centres have experienced unprecedented growth serving as the digital infrastructure of the modern world, further being driven by the ever-expanding digital ecosystem. All this has resulted in the rise of their energy consumption, contributing to the carbon footprint and environmental challenges. This thesis highlights the pressing necessity to optimise energy consumption and minimise the environmental impact of modern data centres.

The aim of this master's thesis is to highlight the challenges in energy-efficient management of operations within a data centre environment. Given the complexity originating due to heterogeneity and the vastness of the components in a data centre (DC), there should be a fine-grained study and evaluation of the impact of various components ranging from the infrastructure layer to the data centre hardware layer, underlying software and the applications running on them. The HPC data centre ontology is crucial in providing a structured and standardised way to represent the complex and dynamic environment of HPC clusters like DAS-6. This report proposes an ontology model for HPC clusters based on metrics collected from two production clusters from HPC data centres, and proposes a simplistic design of an ODA framework (driven by the modelled ontology) for analysing energy-efficiency-related and anomaly-detection-related aspects of a DC processing various kinds of workloads (either scientific or industrial). Further, the same approach is implemented as part of 'ODAbler', an ODA framework built as a digital twin of OpenDC, and energy-related experiments are carried out to validate the energy efficiency and performance goals as desired.

In conclusion, the thesis underscores the pivotal role of an ODA framework following the design of an ontology-driven approach by offering actionable insights and recommendations that would be helpful for data centre operators, policymakers, and stakeholders to advance a greener, more sustainable future.

Acknowledgement

This work could not have been successfully accomplished without the help of many people.

Thanks to Alexandru and Xiaoyu for their excellent supervision and support. It has been a pleasure working with them, most especially my primary supervisor, Alexandru, who constantly gives his support in all of my endeavours throughout my studies. I have enjoyed working with the 'AtLarge Research' team and look forward to collaborating with them even after graduation.

Thanks to Xiaoyu, our doctoral research fellow in the team, for all the help and support, especially for the guidance about the ontology design and data analytics that I used in this thesis project. I have learned a lot from her - both technical and personal.

Thanks to the whole 'AtLarge Research' team for warmly welcoming me into the discussions, and for all the assistance when I was starting.

Thanks to my family for their love and full support in all of my endeavours. Thank you for helping me with my finances, especially during the time that I was having financial difficulties.

Thanks to all of my friends (and 'best' friends) in India and the Netherlands for keeping in touch with me and for making me feel like I'm still home every day through our constant messages.

Thanks to my closest VU friends ("13th-floor folks") for the good friendship and company. I hope we can still keep in touch in the coming years and keep hanging out every time we get 'lost' and feel stressed out with our research.

Special thanks to my former co-workers from 'Keylane' (The Netherlands) and from 'State Bank of India' (SBI), for all the unending and unconditional support and for always keeping me sane, especially during my downtimes. Thanks for believing in me and showing me how proud you are of me. You both are a big part of my success and of who I am today.

Lastly, special thanks to my well-wishers for all the countless blessings that you have been showering me.

This is all for you, lovely people!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The complexity of modern large-scale computing infrastructures (e.g., HPC clusters, supercomputers, and cloud systems) has grown to an extreme level, at the verge of exascale, which introduces operational challenges. The complexity of these systems originates due to their adoption of heterogeneous architectures, ability to provide support to modern workflows and other applications, novel cooling mechanisms, modern infrastructure facilities, and several other components [1]. These systems typically have thousands to millions of CPU cores running up to a billion threads involved in complex computation. Moreover, the dynamic nature of workloads involved in these environments adds to the complexity. All these complexities result in understanding the state of such systems significantly challenging, before even talking about the optimal decisions involved in their operation. It is, unfortunately, common that these systems generate various kinds of malfunctions many times per day, resulting in process crashes ranging to even halting of operations on the compute nodes. Resilience has been studied for HPC executions on these future exascale systems, and several technical options have been established [2]. However, this exascale resilience problem is far from solved, and thus the associated research problems pose a critical challenge for the HPC community.
Also, as the architecture of HPC has evolved over the years, so has the energy consumption of these supercomputers [3]. Due to the expanding use of HPC systems, energy consumption is also expected to expand, causing sustainability-related concerns (for example, CO2 emissions). This needs to be taken into consideration, otherwise, it will impact heavily on climate change. Safeguarding the interests of the community is a journey, which requires a multidisciplinary team of scientists, engineers and technologists to collaborate on this research. Several initiatives are being taken in this regard to meet the organisational energy efficiency goals, and the supercomputing community is promoting the same and publishing the greenest HPC supercomputers to create awareness[4].

1.1 Context

We set the preface of this project by asking a pair of hypothetical questions: "If we were to extract efficiently and quickly a significant bit of information in a large-scale computing system, what kind of decisions would we be able to make? Would adding this intelligence to the system change the way we design software operating on large-scale infrastructures like HPC or cloud?". Of course, some of the possi-

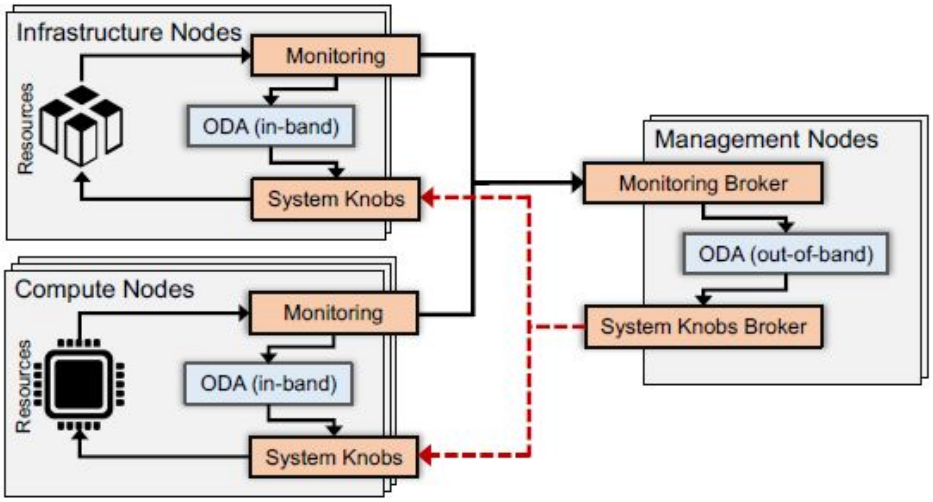


Figure 1.1: Typical data flow involved in a distributed system using ODA (Source:[5]).

ble answers would preface an imagination questioning whether this capability has existed before, and if so, what were the limitations. This project aims to deeply dive into the journey to answer such questions. The first known practical usage of performance dashboards in IT operations began around 2000 when the field of business intelligence (BI) converged with performance management, resulting in the creation of the term "performance dashboard" [6]. Dating back to the 1980s, executive information systems (EISs) were built for the company executives to serve as executive dashboards for driving the companies by their respective bedrooms, but they never gained much traction because they were available to a few people in the company and were built on mainframes or supercomputers (which gradually gave way to client/server systems in the 1990s). The dashboards were already in use in automobiles and other vehicles back then, but businesses, governments, and non-profit organisations have been known to adopt them later.

Large-scale computing infrastructures like supercomputers, clusters, and clouds are already pervasive as most of the members of our society interact with them on a daily basis, e.g., social networks, media streaming services, government services, etc[1]. The purpose of Operational Data Analytics (ODA) is to gain insight into the behaviour of such large-scale computing infrastructure (like HPC clusters) by analysing the operational data from various layers of the computing system [1]. The ODA-related analysis yields various interesting results and behaviours about these large-scale systems which one could not even be aware of. The typical data flow involved in a large-scale system having ODA enabled is shown in Fig. 1.1 [5]. One of the most prominent large-scale infrastructures where ODA-related research has been done extensively is the HPC cluster. High-performance computing (HPC) refers to a specialised branch of computing that has the ability to solve advanced computations (which are too large to be solved using commodity computational resources) with the help of supercomputers or computer clusters. As per one of the tech reports published by a technical college, *HPC integrates systems administration (including*

[1] `https://www.universiteitleiden.nl/en/science/computer-science/systems-and-security/large-scale-computing-infrastructure`

network and security knowledge) and parallel programming into a multidisciplinary field that combines digital electronics, computer architecture, system software, programming languages, algorithms, and computational techniques[2]. HPC architectures couple together powerful integrated compute nodes using a high-speed interconnect. In our study, we are mainly concerned with large-scale computing infrastructures that run exclusively a variant of POSIX-compliant Linux operating systems (OS). The OS on each compute node runs various services and specialised hardware drivers that allow the applications to utilise the resources that are distributed across several nodes.

There are impacts at various layers and scales of the large-scale infrastructures ranging from facility and hardware to the software and applications running on these systems. One of the interesting studies measured the energy efficiency of 27 popular programming languages, measuring the correlation among energy, time and memory [7]. There is a lot of research work ongoing in the related fields due to rising interest in our community to explore the topic in a holistic manner (some of which are discussed in this literature). The term "ODA" was officially defined for data centre operations' monitoring and analytics in 2019 by Bourassa et al. [1]. Its popularity has been rising gradually (although slowly) as can be inferred from the increased count of published works of literature related to the topic. In contrast, data centre monitoring has been there for a long time, exhibiting almost a constant trend. Data centre monitoring might appear overall to be less actively researched compared to other trending computing terms, though the data centre components/entities are being actively researched either individually or in comparatively smaller conjunctions, and not from a holistic perspective (as what is the goal of ODA). As part of the manifesto of Future Computer Systems and Networking Research in the Netherlands, we aim to create awareness about the ODA framework within the Netherlands to promote the implementation of those capabilities across the entire information and communications technology (ICT) infrastructure [8].

1.2 Problem Statement

Quoting the vision of "Future Computer Systems and Networking Research in the Netherlands" [8]:
The grand challenge in the ICT environment of the Netherlands is "to make system-wide qualities easy to observe, reason about, and manage optimally". The fundamental research challenge includes enabling Operational Data Analytics (ODA) across the entire ICT infrastructure.

A lot of research is done every year aiming to promote sustainability and energy efficiency goals for the ecosystem. ODA technique aims to achieve energy efficiency by ensuring that the data centre operations are optimised, along with the business goals being met successfully. As quoted above, it is not easy to reason about the events and their outcomes in an HPC environment without having sufficient access to the underlying states of these events. Hence, a need arises to observe the event, reason about the same using data captured at various levels, and optimise the oper-

[2] `https://web.archive.org/web/20100731043053/http://system.tstc.edu:80/forecasting/techbriefs/HPC.asp`

ations by continuous monitoring and optimising using Operational Data Analytics. Before one starts designing a standard ODA framework for analysing the HPC environment, it is important to understand the ontology model providing a unified representation of HPC resources. There is a need to propose a unified ontology which is comprehensive enough, that could model the publicly available datasets consisting of cluster metrics collected from the HPC environments.

Also, once the ontology model is available for the HPC environment, there is apparently a lack of an ODA framework for data centre simulators which could enable easier experimentation with the datasets (e.g., traces, performance metrics), allowing for extensive research in this regard. There are works done in pieces related to energy-efficiency analysis on cloud environment simulators, but there is currently no published literature which discusses the data centre operations and its ODA capabilities as part of an ODA framework separately.

1.3 Research Questions

The main objective of this thesis is to create a proof-of-concept ODA framework for large-scale computing infrastructures like HPC, cloud, etc. Such a framework would address the problem highlighted in Section 1.2. We consider the size of the DAS-6 cluster hosted at VU as the base size of the cluster for all of our ODA-related experiments[3]. Here we list the research questions covered as part of the scope of this thesis project. These research questions will be answered subsequently in the upcoming sections of this study.

- **RQ1** (HPC cluster-metrics-based ontology modelling): **How to design the ontology of a large-scale computing infrastructure (typically HPC cluster) using the metrics exported in a time-series format?**
  The answer to this research question stands as a novel contribution as part of this thesis report. This research question will conceptualise a novel way of designing ontology for large-scale computing infrastructures like HPC sites, with a focus on modelling data from the metrics exported in a time-series format.

- **RQ2** (Designing an ODA framework): **How to design an ODA framework for a data-centre simulator like OpenDC to realise some of the benefits of the ODA techniques?**
  This section will be the major contribution as part of this thesis report. We will briefly discuss the architecture 'ODAbler' - the ODA framework we have designed for OpenDC based on certain requirements.

- **RQ3** (Evaluation of the designed ODA framework): **What are the various (quantitative/qualitative) energy or performance benefits that have been realised as part of experimental evaluation of the ODA framework?**
  In this section, we will go through the experimental evaluation of the ODA

[3]DAS-6 cluster @ VU: `https://www.cs.vu.nl/das/clusters.shtml`

framework 'ODAbler' configured for a large-scale distributed data centre simulated in OpenDC when various workload traces are executed in such an environment with different cluster-size configurations. We will talk about the gains realised in quantitative terms (related to the energy-efficiency analysis of a set of task-scheduling algorithms) and/or qualitative terms (in terms of anomaly detection), wherever applicable.

- **RQ3.1 (Demonstration of an energy-awareness scheduling analysis):** How to validate the energy-awareness benefits realised (if any) after enabling an ODA framework for a large-scale computing infrastructure?
  With this research question, the idea is to present the out-of-band ODA analysis for the best energy-aware task scheduling algorithm among the available task scheduling algorithms for resource managers, as this topic is the crux of energy efficiency research. This analysis would pave the way for further exploration of mapping specific task scheduling algorithms while executing a specific nature of trace against a specific cluster scale and configuration, based on the learning obtained from past experimentations.
- **RQ3.2 (Demonstration of node-related performance-leak detection analysis):** How to validate the node-performance-related benefits realised (if any) after enabling an ODA framework for a large-scale computing infrastructure?
  With this research question, the idea is to present an in-band ODA mode covering online analysis that presents the way the faults or anomalies are detected on a node level to begin with, as this field is gaining momentum gradually (related to the field of performance engineering). The node-level anomaly detection would help the system engineers figure out the performance leak (covered in this report), and fix the problem (out of the scope of this project), thus contributing to improved performance.

1.4 Research Methodology

The research methodology involving this thesis is discussed below.

- **RM1 - Standard Literature review process (SLR)**: An extensive literature study was conducted (forming the basis of this work) to study the various ODA frameworks operational in various large-scale computing infrastructures. That study helped in arriving at the 'ODAbler' design based on the proposed reference architecture for the ODA framework.

- **RM2 - Design, Prototyping, and Evaluation**: ODAbler has been designed and evaluated as part of this master's thesis. The project has been made possible with the help of literature study work, and adapting to various energy usage analysis related research already done on the OpenDC simulator.

- **RM3 - Experimental research (energy-efficiency analysis)**: This experimentation is inspired by the learning from a master's course on Distributed Systems lectured at VU (by the head supervisor of this thesis). The previous project done as part of the course was to compare various task scheduling

algorithms in terms of various parameters related to makespan and energy usage. This work inherits the implementation of task-scheduling algorithms like HEFT, MinMin, etc. from that project[4]Previous experience in the OpenDC project - `https://github.com/CloudScheduling/opendc`).

- **RM4 - Experimental research (node-anomaly analysis)**: This experiment has been selected for ODA analysis based on various works of literature and standalone thesis projects based on anomaly detection. The 'Background' chapter discusses anomaly detection and related work in detail.

1.5 Thesis Contributions

We aim to provide the following contributions to the community as part of this project, which would hopefully add to research in the field of Operational Data Analytics (ODA):

- **C1** - Highlighting the benefits of modelling ontologies from operational data, which could enable automated reasoning about relationships between different concepts of large-scale computing infrastructure(s).
- **C2** - Contributing to the design of an ODA framework for a data-centre simulator (OpenDC) which could facilitate the experimentation of various models for testing operational efficiency (both energy and performance efficiency).
- **C3** - Conducting experiments to determine the energy efficiency among various task scheduling algorithms used in the data-centre simulator for workload execution, highlighted by the designed ODA framework.
- **C4** - Conducting experiments to identify the node anomaly related to power usage, highlighting any performance leak concerns using the designed ODA framework.
- **C5** - Discussing the design of the 'ODAbler' ODA framework, which facilitates the operation of a data centre simulator with enabled experimentation related to data analysis, without significantly affecting the operation or functionality of the former.

Novel Contributions

We claim the following contributions from this project which are conceptually novel, as they are being explored for the first time in the research field:

- **NC1** - Representation of an ontology model derived from time-series data collected from multiple HPC clusters. To the best of our knowledge, we have not found a mapping process of ontology modelling based on collected cluster metrics in time-series format.
- **NC2** - Contribution to the design of an ODA framework for a data-centre simulator enriched with the module that facilitates the experimentation of various ODA functionalities for driving operational efficiency. We have not found any work of literature detailing ODA framework usage in a data centre simulator yet.

[4](

- **NC3** - Contribution to the easier experimentation in OpenDC using an ODA framework for a data-centre simulator, which adheres to the FAIR principle, discussing various ODA modes (in-band, and out-of-band) of operation and ODA techniques (online, and on-demand) in use. In addition to the contribution of an ODA framework, the 'ODAbler' analyser is flexible with experimentation capabilities, which is one of its kind for a data centre simulator.

Societal Relevance

We foresee that our contribution might lead to awareness in society about sustainability practices, promoting the adoption of energy-efficient methods for managing day-to-day operations. This would have a far more significant impact on the modern era, where there is a lot of concern involving climate change due to sub-optimal practices in different aspects of the industries.

1.6 Plagiarism Declaration

This project report was written by me and in my own words, except for quotations from published and unpublished sources which have been clearly indicated and acknowledged as such. I am conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism, subject to the custom and usage of the subject, according to the University Regulations and Guidelines regarding examinations. The source of any picture, plot or other illustrations is also indicated, as is the source, published or unpublished, of any material not resulting from my own experimentation, observation, or specimen-collecting.

1.7 Open Science

Abiding by the FAIR data principles, all of the work revolving around the ODAbler project (including OpenDC integration) has been made available in their respective GitHub repositories. The 'ODAbler' repo is comparatively richer in terms of a detailed guide for reproducibility, and includes documentation about the work done in terms of video tutorials, making available the former literature study work for deep diving into the ODA research. The detailed guide to the replication of the work done under this thesis is discussed in Appendix A and Appendix B respectively.

1.8 Thesis Structure

The remainder of the thesis is structured as depicted in Fig. 1.2. We start with a brief overview of the key concepts revolving around the subjects in Chapter 2. Chapter 3 discusses the HPC ontology modelling based on two of the industrial datasets collected from HPC environments, facilitating the understanding of various metrics generated by existing monitoring/ODA frameworks that are enabled in production, and paving the way for designing one such framework for a data centre simulator environment 'OpenDC'. In Chapter 4, the design of 'ODAbler', a simple ODA framework for OpenDC, is presented discussing the design choices and underlying decisions. We evaluate the features designed as part of the 'ODAbler' project for OpenDC in Chapter 5, and present some interesting results demonstrated by

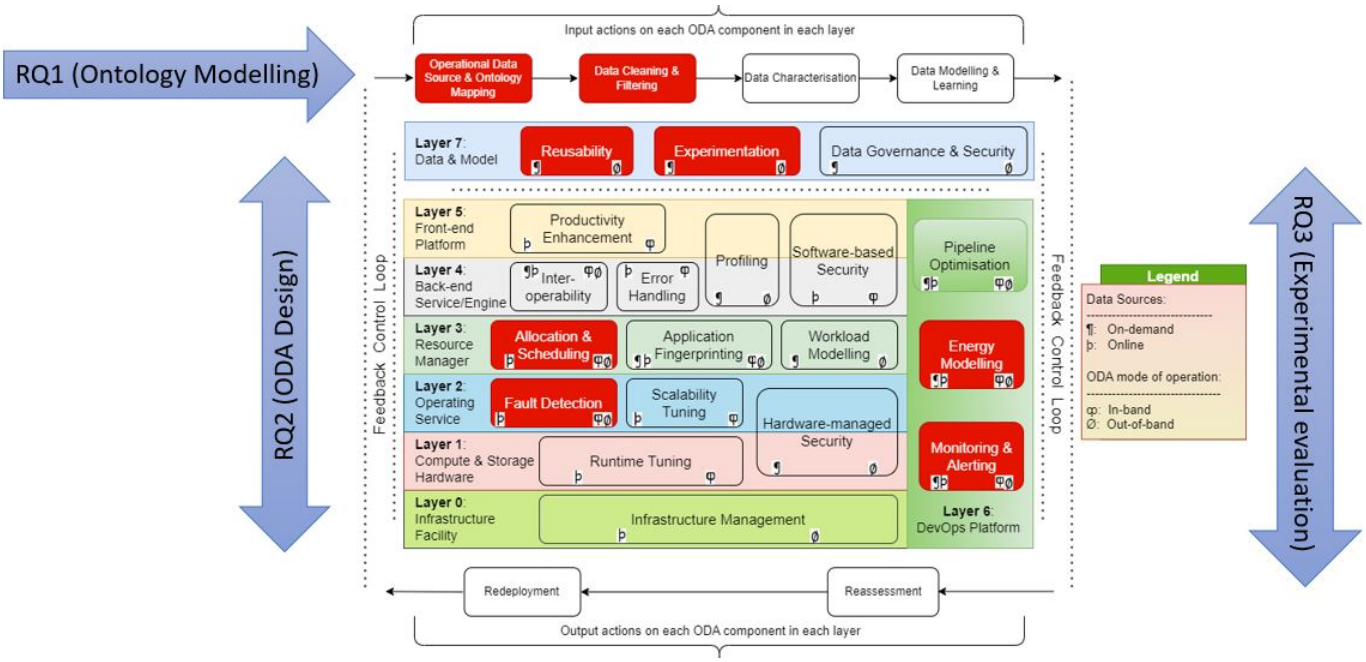


Figure 1.2: Thesis overview

the framework, thus validating the operational data analytics-related claims. Finally, in Chapter 6, we summarise the contributions of this thesis and present future directions for the project to promote awareness of ODA-related benefits in the community. Additionally, an appendix is attached which links to the source code and artefacts available resulting from this thesis, as part of the FAIR initiative adoption.

Chapter 2

Background/Literature Review

In this section, we discuss some of the underlying concepts which form the backbone of this thesis project. There is an additional section on the design process, which is strongly encouraged for projects that are associated with the "@Large Research" team. This project also adheres to the corresponding design process and principles.

2.1 Large-scale Computing Infrastructure/Ultra-large-scale systems (ULSS)

Large-scale computing infrastructure, or Ultra-large-scale systems (ULSS) term is used in the field of computer science to refer to software-intensive systems having the influence of unprecedented amounts of hardware, users, volumes of data, and lines of code. In terms of computing infrastructure, it includes the pervasive ones with which we interact directly or indirectly on a regular basis - supercomputers, clusters and clouds, while using social networks, media streaming services, government services, etc[1]. All these are powered by such large-scale infrastructure. A generic model for the operations of such large-scale computing infrastructure is laid out in Fig. 2.1 [9].

HPC: High-Performance Computing

High-performance computing is the ability to solve advanced computation problems with the help of supercomputers or computer clusters. As mentioned by TSTC in their technical report[2], *HPC is a multidisciplinary field, combining digital electronics, computer architecture, system software, programming languages, algorithms and computational techniques in parallel computing environments to model complex systems or to control transactional computing services.* HPC is applied to many different technologies and it is rapidly evolving. HPC was most frequently associated with scientific research in the past; however, it is being applied to business, government and military uses of cluster-based computing strategies, such as data warehouses, line-of-business (LOB) applications, transaction processing, utility computing and

[1] Large-scale computing infrastructure - `https://www.universiteitleiden.nl/en/science/computer-science/systems-and-security/large-scale-computing-infrastructure`

[2] HPC - `https://web.archive.org/web/20100731043053/http://system.tstc.edu:80/forecasting/techbriefs/HPC.asp`

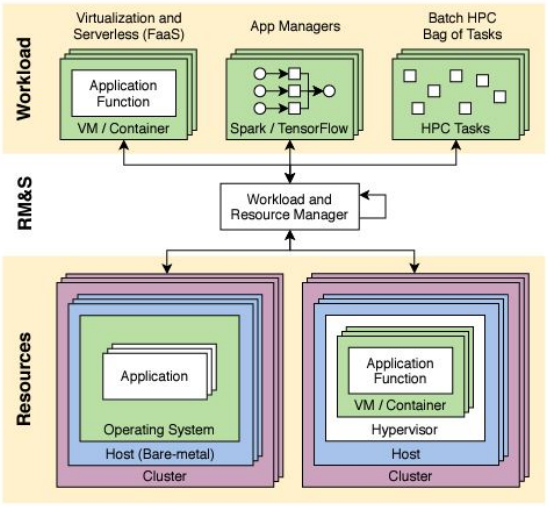


Figure 2.1: Generic model for data centre operations (Source:[9]).

next-generation telecommunications.

Cluster computing

HPC clusters link multiple nodes through a LAN (local area network), and are uniquely designed to solve one problem by spanning it across the nodes in a system. These interconnected nodes act as a single computer.

Grid and Distributed Computing

Grid computing and distributed computing have synonymous architectures. They constitute multiple computers connected through a network, sharing a common goal to perform a large computational task. Each node can perform tasks independently without communicating with other nodes in the system. Grid computing distinguishes itself from conventional distributed computing in terms of its focus on large-scale resource sharing, and high-performance orientation in some cases. The core difference between them and cluster computing is that the former has nodes connected in a heterogeneous network, whereas the latter (cluster computing) has a homogeneous network of connected hosts.

2.2 Data Centre Simulation

Simulation is defined as the "imitation of a real-world process or system over time, enabling the study of, and experimentation with the internal interactions of complex systems" [10]. We consider a specific type of simulation, *"discrete-event simulation"*, where a system's operation is represented as a sequence of events over time, in addition to the assumption that no changes occur in between events. This type of simulation allows considering only the direct progression of events, unlike the continuous models. Almost all efforts to model cloud and data centre operations rely on discrete-event simulation, owing to the sheer scale and complexity of the data centre operations and the nature of experiments [11].
The data centre simulation application provides the functionality of a virtual model representation of the data centre and external infrastructure components like generators, cooling towers, etc. that facilitate various tasks like energy-consumption-

related simulations, performance-related experimentations, etc. Various popular data centre simulators have been used in academic research in the past, some of which are briefly discussed below. Please note that our work is based on one such data centre simulator called 'OpenDC' for multiple reasons which are discussed in the corresponding subsection.

Reasons for selecting a simulator for ODA-framework design (OpenDC)

It is true that experiments involving physical infrastructure deliver results closer to real-world operations. However, the main challenge in such large-scale computing infrastructure involves the reproduction of problematic scenarios that would result in the significant exploration of the problem space (due to various phenomena, e.g., network variability [12]). Other relevant challenges associated with real-world experiments on data centres are that they are time-consuming, expensive, and have a lot more impact on the environmental footprint.

On the other hand, simulation enables the flexibility to investigate a wide range of "what-if" scenarios. In our current project, the simulation coupled with an ODA framework (discussed later) allows us to represent the data centre environment model virtually, and analyse the power usage and energy consumption of implemented task-scheduling algorithms, helping us understand if there is a single winner in terms of the most energy-efficient algorithm. Another experiment based on simulation allows node-anomaly analysis based on the power usage metrics of the cluster environment, helping identify if there is any related anomalous behaviour on any node. These experiments have been validated as reproducible (with various tests included in the simulator), involve almost negligible cost as the experimentation is carried out on a professional laptop, and also contribute to extremely low energy cost and environmental footprint comparatively. Thus, a data centre simulator (OpenDC) has been selected as the source for designing an ODA framework (ODAbler), which acts as the "Digital Twin" of the former. A **digital twin** is "*a virtual representation of a physical system (and its associated environment and processes) that is updated through the exchange of information between the physical and virtual systems*" [13]. In simpler terms, ODAbler virtually models the efficient operations of OpenDC and, thus, is coined as the digital twin of OpenDC.

CloudSim

CloudSim is a high-quality data centre simulator focusing on the simulation of cloud system components including virtual machines, data centres, and resource provisioning policies [14]. There are several other single-feature simulators based on CloudSim such as iFogSim [15], WorkflowSim [16], and CloudAnalyst [17] that extend it in a specific field like fog, edge, cloud, etc.

SimGrid

SimGrid is a simulation framework which enables the simulation of distributed applications in distributed computing environments for the specific purpose of developing and evaluating scheduling algorithms [18]. It has been designed to solve the concerns related to accuracy and scalability, and the goal of versatility involving various use cases. However, this has limited scope and features, as compared to other simulation frameworks discussed in the next section.

Project	Environment	Stakeholders	Highlighted Feat.	GUI
CloudSim [7]	Cloud, Edge, Fog [45]	Research	VC⋆, N, S, E, WF† [46], FD†, EXP†, CM, PI†	✓† [47]
SimGrid [8]	Grid, Cloud [48], P2P	Research, Edu. [49]	VC⋆† [50], N⋆, S, E⋆, WF⋆ [51]	✓† [51]
DGSim [52]	Grid	Research	WF, F, EXP	✗
GroudSim [53]	Grid, Cloud	Research	WF, CM, F	✗
iCanCloud [9]	Cloud	Research	VC, N⋆, S, CM	✓⋆
OpenDC (this work)	Cloud	Research, Education	VC⋆, N, S, E, CM, FS⋆, ML, WF, F⋆, PI, EXP⋆	✓⋆

Models: VC = VMs and containers, N = Network, S = Storage, E = Energy, CM = Cost models, FS = FaaS, ML = Machine learning, WF = Workflows, FD = Federation; **Phenomena**: F = Failures, PI = Performance interference, **Tools**: EXP = Experiment automation. **Support**: ✗= No, ✓= Yes.
† = extension, not integrated; ⋆ = advanced, carefully calibrated feature.

Figure 2.2: Comparison of various data centre simulators, as conducted by the authors of OpenDC 2.0 literature (Source: [11]).

OpenDC

OpenDC is a data-centre simulator [11] managed by the "AtLarge Research" team whose first version was released in 2017. The current version 2.0 underwent a radical change involving re-design and re-engineering efforts. The three main high-level components of OpenDC are: (i) a web and textual frontend, (ii) a model-driven discrete-event simulator, and (iii) a set of tools to assist with simulation. This thesis project involves the use of the simulation component and the convenience tools (experiment runner, workload processor, environment processor, and library of metrics). In addition to the features found in CloudSim and SimGrid, OpenDC has a couple of strength points (discussed in the next subsection comparing the three). The detailed architecture and other important details are discussed in Chapter 4 on ODAbler. Further details about OpenDC can be referred to in the linked literature.

Comparison of notable data centre simulators

There are many notable high-quality data centre simulators built by the community like DGSim [19], GroudSim [20], and iCanCloud [21] which have not been discussed here for brevity reasons, but they have been contrasted against each other by OpenDC authors in their literature, as shown in Fig. 2.2 [11]. Referring to the same figure, it can be noticed that other high-quality simulators typically provide a single feature, or are comparatively general with respect to the functionalities available in OpenDC. The closest alternative to OpenDC in terms of functionality is the CloudSim platform, but because of several other extensions extending its capability in a specific research direction, OpenDC has been preferred due to its integrated approach and specific modelling features already discussed in the OpenDC 2.0 literature. Another excellent cloud simulation framework not discussed here is CloudSim Plus [22], which has several modern capabilities worth exploring. Thus, this project

work has been restricted to using OpenDC (as the data centre simulator), given the scope of the project. And, accordingly, ODABler (an ODA framework) has been designed (a novel contribution to this research project) that acts as a digital twin of OpenDC.

Task scheduling simulation

The data centres receive 'workloads' to execute as part of day-to-day operations. These workloads comprise streams of 'jobs' which themselves comprise several 'tasks'. The component 'scheduler' manages the data centre resources, and is responsible for the execution of the incoming workloads. For a task scheduler to schedule a set of tasks, it needs to determine an order of execution and assign each task to a suitable processor(s) [23]. Thus, it needs to consider any task inter-dependencies and service level agreements [24] (if enforced). There are a variety of techniques proposed to find a solution to the NP-hard task scheduling problem [25], and below we discuss a few of them which have been used in the project for experiments. Please note that we have used invariably either the term task scheduling policies or task scheduling algorithms in many places in this report, and they correspond to task-scheduling techniques in general.

FIFO: SubmissionTimeTaskOrder (STTO) or FirstInFirstOut (FIFO) is a task scheduling policy that selects the task from the list of tasks ready to be scheduled in terms of ascending order of their submission time (analogous to first come, first served). This policy also exists by default in OpenDC's original implementation, and, thus has been reused in this project.

HEFT: *Heterogeneous Earliest Finish Time* (HEFT) [23] is a heuristic algorithm that schedules tasks in a way that reduces their finish time, hence it optimises for a short makespan of the whole workflow. HEFT consists of two phases. In the *task prioritising phase*, the execution order is determined by sorting the tasks in decreasing order by their *upward rank*. The upward rank of a task t_i is recursively defined as

$$rank(t_i) = \overline{w_i} + \max_{t_j \in succ(t_i)} (\overline{c_{i,j}} + rank(t_j)),$$

where $\overline{w_i}$ is the average computation cost of t_i, $\overline{c_{i,j}}$ is the average communication cost, and $succ(t_i)$ the set of tasks that depend on t_i. Because of its recursive definition, this ordering satisfies task-precedence constraints. The original HEFT algorithm considers the cost of communication when there is a data dependency between two tasks that are executed on different machines, but in the experiments described in this report, the communication cost was assumed constant as zero (due to the limitation of our data-centre environment). This algorithm's implementation has been reused here from a previous project work done on OpenDC[3], during the distributed systems course at VU.

Random:*Random* is another scheduling algorithm that selects a random task from the list of tasks that are ready to be scheduled on a suitable processor(s). Its implementation already exists in the OpenDC simulator's original work, and thus has been used in designing the experiments.

[3]Previous OpenDC work by the author - `https://github.com/CloudScheduling/opendc.git`

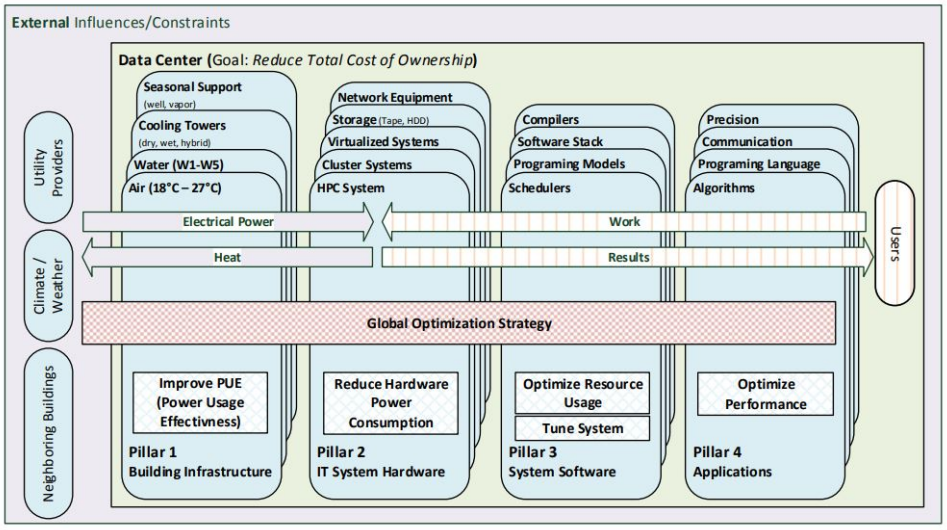


Figure 2.3: Four pillars of energy efficient HPC (Source: [26, 27]).

2.3 ODA: Operational Data Analytics

Bourassa et al. [1] define Operational Data Analytics (ODA) as a mechanism to analyse the operation of an HPC system to gain insight into its behaviour. Several monitoring frameworks capture data from a large number of sensors (located across the infrastructure down to the compute node level) in hardware and software units at fine granularity. The ODA framework establishes a built-in control loop, which enhances the HPC operations by increasing system reliability and energy efficiency among other aspects. Alessio Netti and his team at LRZ (supercomputing centre) along with a few other members involved in ODA projects (Oak Ridge National Laboratory, Hewlett Packard Enterprise, and Energy Efficient HPC Working Group) proposed a conceptual framework for HPC Operational Data Analytics (ODA) [27]. They mentioned the four pillars of energy-efficient HPC (shown in Fig. 2.3) and highlighted the four types of data analytics possible on each of these four pillars (shown in Fig. 2.4).

We present and discuss below a reference architecture for an ODA framework in Fig. 2.5 which is mapped to ecosystems of distributed computing (devised during the formerly conducted literature study). The components in the proposed reference architecture of the ODA framework are complementary to the role handled by the respective layers. It must also be noted that although almost all of the functionalities can be automated and optimised, there might be some scenarios where manual intervention would be required (e.g., verification, validation, etc.), which we do not discuss here. It might vary on a case-to-case basis, and the same is open for interpretation by the researchers who work in this field.

The relevant components of the ODA framework outlined in the reference architecture are further defined below. Please note that various components at different layers in the proposed reference architecture overlap to some extent to contribute to power usage optimisation, energy efficiency, or performance. An additional point to consider is that the target users responsible for managing those components must focus on selective functionalities, giving freedom for management, research and ex-

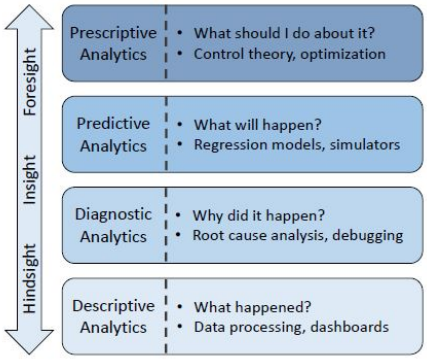


Figure 2.4: Types of data analytics for HPC (Source: [28]).

ploration in detail (they could either be distributed in different teams as described above as key actors or might be from the same team but with slightly different roles). We think that this would enable high productivity, with ease of management and focussed research in the operations, and also refer to the works of literature resulting in the identification of these key functionalities.

Proposed holistic ODA framework (result of the literature study)

The input actions on each of the ODA components first involve the understanding of various metrics available for observation, which can be modelled using **ontology mapping** of the data collected from operational data sources. The data from the data sources may not be used directly as such, and usually some sort of **data cleaning** and **filtering** is required which might be distinct for distinct ODA functionalities as per the customary requirement. Once the data is available after cleaning, **data characterisation** is needed for summarising the general characteristics, after which the designated **modelling** and/or **learning techniques** is applied with a defined objective resulting in an outcome. Each of the ODA functionalities discussed below keeps undergoing **reassessment** and **redeployment** (usually at periodic intervals but also dynamically as and when needed) which is fed back via the **feedback control loop**, for continual optimisation of the data centre operations (as shown in Fig. 2.5):

- **Infrastructure Management**: Being operational at the lowermost building block (at the facility level), this ODA functionality relies on the online data source collected from various sensors from the facility and provides an out-of-band mode of service for managing the infrastructure.

- **Runtime Tuning**: An ODA framework can be used to predict the behaviour of various components of the compute and storage hardware, and perform the dynamic tuning using system knobs. It relies on real-time data and information about specific components supplied from various layers, which is then sent to the compute nodes for dynamic adjustment of the values of those components.

- **Hardware-managed security**: This is a critical security-related functionality which is a part of both compute hardware and operating service layers, and is responsible for assisting in the security of the underlying computing hardware with the help of on-demand data getting tracked with periodic intervals, thereby ensuring that the hardware is not tampered with. This idea is slightly novel due to the fact that there is not much security-related context

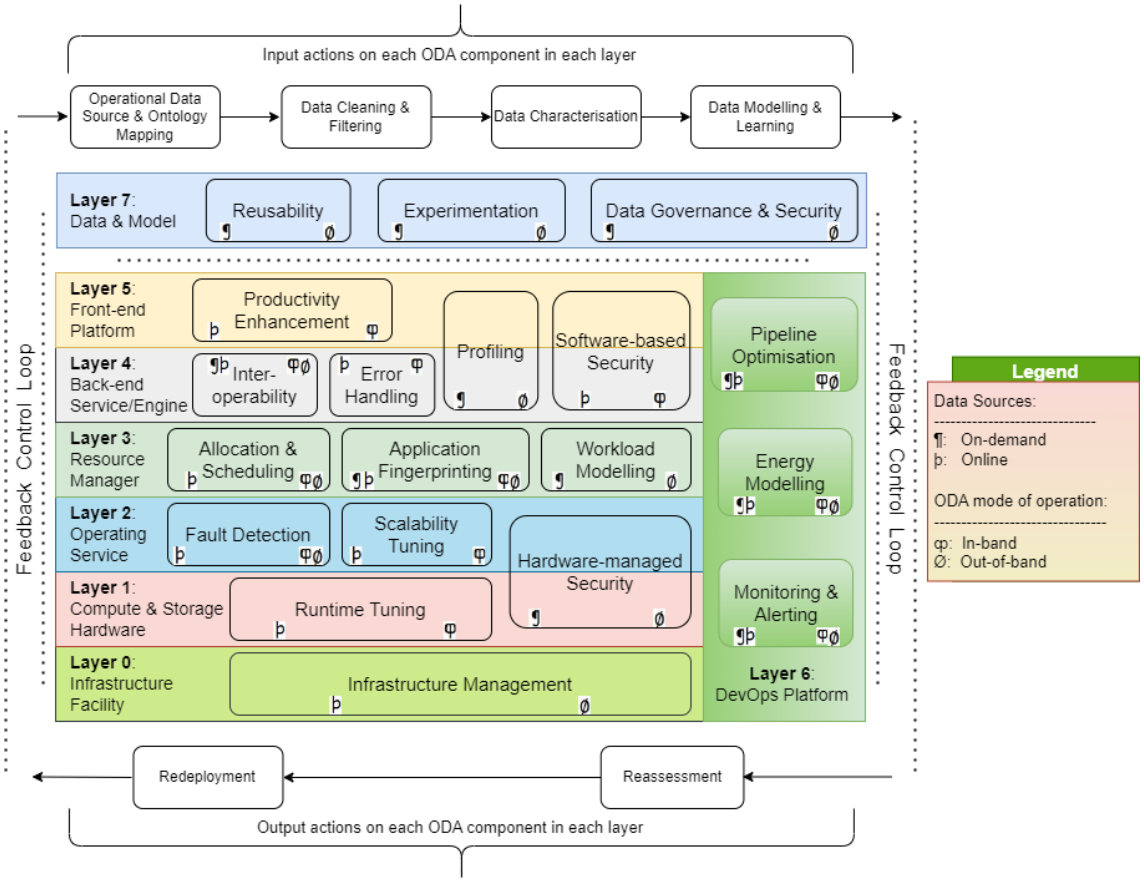


Figure 2.5: Proposal of a holistic ODA framework in previously conducted literature study

made available in the state-of-the-art ODA frameworks (except LBNL's newer OMNI architecture deployment which discusses some traits about the security considerations handled within the ODA framework).

- **Fault Detection**: This ODA component is responsible for detecting and predicting anomalous states in hardware and software components, thereby helping to improve the resiliency of the system as a whole by helping prevent catastrophic events. This monitors data from underneath as well as top layers on a real-time basis and helps detect and predict faults in both hardware and software components (continuously as well as periodically reviewing throughout the sections), preventing in turn unmasked failures.

- **Scalability Tuning**: It is crucial for the operating services to closely monitor the scalability of the compute and storage resources (physical or virtual) as per the requirements of the jobs. Thus, the ODA framework must use real-time (online) data from the cluster manager and related computing layers.

- **Allocation & Scheduling**: As crucial as something could be, this is the backbone operation of the resource manager and is responsible for the optimal placement of jobs or allocation of resources. This needs real-time data about jobs and resources, supplies additional information to the scheduler for achieving the ODA goals, and operates in both in-band and out-of-band modes.

- **Application Fingerprinting**: The idea involves predicting the behaviour of user jobs and correlating this with the data from the past to characterise fea-

tures such as power consumption and memory utilisation, helping to optimise management decisions. This relies on data sources from online sources as well as from the past and operates in both in-band and out-of-band modes.

- **Workload Modelling**: An ODA framework uses heuristic or learning techniques to predict the properties (duration, submission pattern, etc.) of jobs submitted by users and helps in improving the effectiveness of scheduling policies. This relies on data sources from the past and operates in out-of-band mode to help reduce queuing times.

- **Interoperability**: The ODA framework should be able to manage and fine-tune the interoperability requirements with various layers, processing both real-time (online) and historical data while operating in either in-band or out-of-band modes.

- **Error Handling**: This is another crucial functionality which is responsible for monitoring the error events (online) originating from the back-end engine or as reported by other upstream layers, and acting upon them on a real-time basis to provide a reliable service.

- **Software-based Security**: Similar to hardware-managed security, software-based security is spread across both the front-end platform and back-end engine and ensures security by monitoring and acting upon the security events collected online from multiple layers as part of near real-time operation handling.

- **Productivity Enhancement**: The idea at the front-end layer is to enrich the user experience resulting in productivity enhancement. The component could optionally involve seeking custom input from the user or the data source could be the relevant front-end application logs analysed at regular periodic intervals, and desirable action could result in contributing to a rich user experience.

- **Profiling**: The ODA framework should be able to profile the code or the binary used in the front-end platform or the back-end service in an out-of-band fashion, and suggest a suitable action or perform automatic remediation for improving the performance.

- **Monitoring & Alerting**: The majority of the ODA framework functionality revolves around this core functionality of the DevOps layer. Monitoring and alerting functionality relies on data from almost all other computing layers for acting on them, and includes the **visualisation** of the various operational data and/or analysis results to the different segments of users discussed previously. This functionality stands out as one of the most important ODA applications using data from both online sources and on-demand from distinct sources, and operating in both in-band and out-of-band modes.

- **Energy Modelling**: This is an outcome of various ODA functionalities discussed before. The idea involves modelling the energy of a system by employing data analysis for investigating different scenarios or assumptions.

- **Pipeline Optimisation**: The pipelines in the DevOps layer can be optimised by cleaning up the stale data, information or logs, and applying data transformations on the speed of ingress/egress, or on the data formats for compression

and type-conversion. The automatic builds defined as part of DevOps practices can be monitored and acted on efficiently to ensure their acceleration and faster delivery of results, utilising the underlying infrastructure optimally. These must be monitored continuously (on individual components) and at periodic intervals (to have a holistic overview) and should be configured to use real-time data or offline data at the DevOps layer.

- **Reusability**: Some of the data collected at various data sources or the developed models might be reusable in nature (for operational analysis), and it could help in avoiding reinventing the wheel at times. This intelligence could be available to an ODA framework so that it can predict, identify and reuse the data or the models as and when required.

- **Experimentation**: The data sources or the models should allow experimentation for enriching the learning of the entire ODA ecosystem, which can be done at periodic intervals by collecting data from various layers. Some of the similar aspects related to this are testing, measuring and benchmarking.

- **Data Governance & Security**: This layer is pertinent to identifying the data, characterising it, ensuring that they are of high quality, and improving its value (with archival and cleanup policies as desired). It can help organisations to respond to threats without a significant delay. This activity can be conducted periodically on-demand, and should also include the data collected from relevant layers.

This ODA framework has been designed to cover all relevant forms of analytics discussed earlier, and the scalability aspects for any component can be considered during the time of its implementation, to optimise the respective operational goals. Also, we have tried to reuse the concepts from the massivising distributed processing ecosystem literature and energy-efficient pillars of HPC and have tried to build the reference architecture covering them all. We expect that this idea can be utilised in other domains of computing or even other fields to optimise operational efficiency.

Notable ODA frameworks in use

There are currently various ODA frameworks which are in use in production infrastructures, and it is not possible to list all of them. The author has done a literature study[4] before starting this thesis project that delves deeper into detailed specifications of a couple of those notable ODA frameworks, which is also available as a document in the 'ODAbler' project GitHub repo's 'documentation-guide' directory. Some of the notable ODA frameworks which serve as an inspiration for this thesis project are discussed in this section.

1. Operations Monitoring and Notification Infrastructure (OMNI), National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Laboratory (LBNL) or, Berkeley Lab, USA

NERSC utilises multiple monitoring systems for their operational monitoring, amongst which OMNI and SkySpark[5] are the most prominent ones [1]. They have documented an effective use of operational data instrumentation, analysis, integration,

[4]ODA-based literature study: `https://github.com/am-i-helpful/ODAbler/blob/master/documentation-guide/Shekhar_Literature_Study_about_ODA_on_Energy__2023_Final.pdf`

[5]SkySpark, SkyFoundry `https://skyfoundry.com/product`

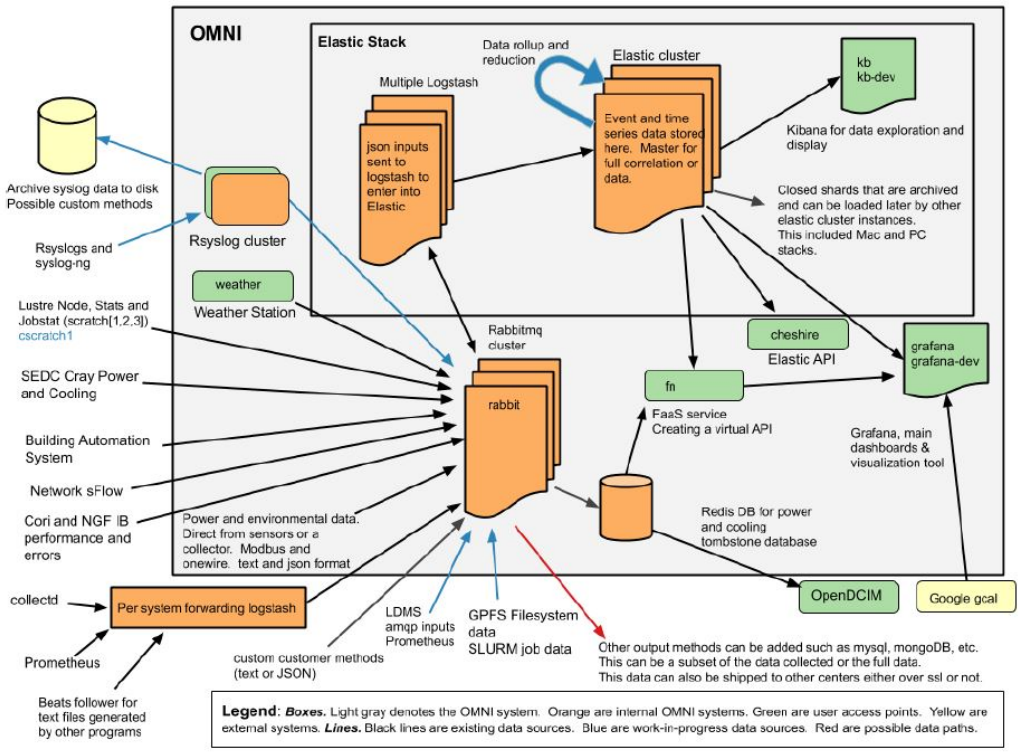


Figure 2.6: OMNI Integrated Operational Data Collection and Analytics Architecture (Source: [29]).

and archiving, that aims towards effective design, commissioning, and optimisation of power usage effectiveness (PUE) in their HPC facility environments [29]. It is worth mentioning that 16 of the NERSC systems appear on the TOP-500 list of fastest computing systems in the world [29, 30]. Their mission is to provide HPC and compute resources to science users guaranteeing high availability along with high utilisation of the systems to bolster the scientific research for the Office of Science in the U.S. Department of Energy (DOE).

Monitoring and Optimisation tools: The architecture of OMNI is shown in Fig. 2.6 [29]. OMNI is a multifaceted platform of applications that combines a large amount of HPC and IT systems data with comprehensive cooling and facility systems performance data. The NERSC OMNI system merges the Building Management System (BMS) data (covering an array of rack-level IT sensors) with Cray syslog data, resulting in a real-time, searchable, and easily visualised ODA system. The resulting Elasticsearch dataset then (at the time of publishing of corresponding literature) archived 25k data points/sec within the general HPSS storage system, which was further planned to expand to 100k data points/sec in late 2019. The graphical visualisation of the real-time data is managed by the Elasticsearch Grafana and Kibana web browser-based user interfaces, whereas various other analysis and visualisation needs are completed using more specific open-source software tools as needed. On the other hand, SkySpark interfaces with building systems (e.g., via BACnet) to collect and analyse available building data. The SkySpark platform gathers data through a live connection to the NERSC BMS, the Elasticsearch database, and an ION power meter database. The NERSC Energy Efficiency (EE) team have built their custom views for performance metrics which are auto-

matically updated with the live OMNI data link.

OMNI design and implementation: OMNI is an integrated operational data collection and analytics infrastructure. The OMNI data collection architecture is shown in Fig. 2.6 along with its diverse data sources. The core system requirements identified for creating OMNI system are scalability (managing the volume of systems and sensor data to provide near-real-time insights), high availability (data collection infrastructure must be always available, even in the presence of some issues at the data centre), maintainability (to apply patches, upgrades, warm hardware swaps, etc. to parts of the system without affecting the flow of data), usability (fast and easy access to the collected data for analytics, visualisation, and monitoring purposes by different actors), and lifetime data retention policy (collecting and saving the data forever theoretically, for statistical modelling and failure prediction from historical data). Accordingly, the OMNI cluster has been made independent of any other system in the facility, which becomes available as soon as the power is turned on (and the last one to be taken down before powering off). OMNI has been implemented with the help of various open-source software, on-premise hardware, and virtualisation technologies, and keeps collecting data as long as there is power to the facility. Virtualisation is implemented using oVirt and Rancher, and data ingestion and storage are managed by Elastic stack in near-real-time. The key component in the elastic stack is Elasticsearch (a distributed JSON-based RESTful search engine) that facilitates real-time ingestion and search within massive amounts of data. The Logstash component manages the server-side data processing pipeline, and forwards them to Elasticsearch for ingestion. The Kibana interface provides a web interface for data discovery, analysis, and visualisation of Elasticsearch data in addition to the monitoring information and management controls for the Elastic stack.
Data is collected from various systems (Systems Environment Data Collections (SEDC) data, job information from Slurm scheduler, Lustre parallel file system data, information from the Aries high-speed network, etc.) and sensors (ranging from building management systems like BACnet, Modbus, and various other facility sensors, to power distribution units (PDUs), Uninterruptible Power Supplies (UPSs) etc.) into Elastic search with the help of RabbitMQ cluster (a popular messaging broker supporting multiple messaging protocols and queuing), which sends it further to Logstash. Logstash acts as the local aggregation point by reducing the number of network connections managed by the central logging clusters, and forwards a single connection to the central logger in an encrypted manner (and could also convert collected UDP packets to TCP to ensure reliability of data). Elasticsearch indexes the data for near-real-time retrieval and querying, which could be either queries using its native RESTful APIs or by visualisation and data discovery tools such as Kibana and Grafana.

2. DCDB (Data Center Data Base) Wintermute, Leibniz Supercomputing Centre (LRZ), Germany

Wintermute is an ODA framework implemented on top of the holistic DCDB monitoring system that offers a large variety of configuration options to satisfy the varying requirements of ODA applications [31]. It enables the analysis of data and granular level control at various levels of an HPC system. The authors originally suggested the use of two types of data sources (in-band and out-of-band), and the

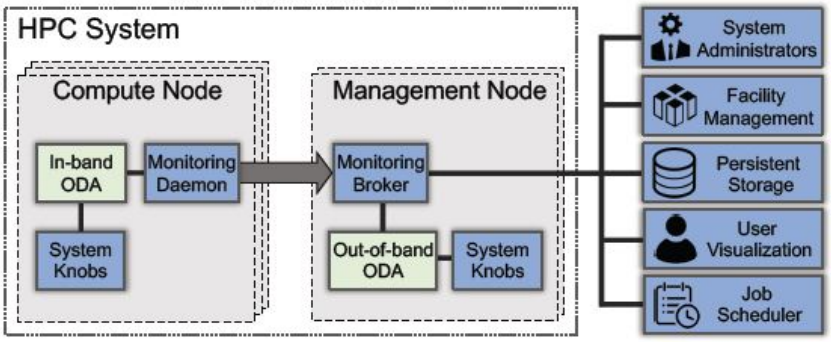


Figure 2.7: Wintermute: A high-level overview of the suggested architecture for an online ODA framework integrated into a monitoring system, showing the main components and actors involved (Source: [31]).

grouping of ODA techniques (into online and on-demand modes), both of which the authors borrowed in their ODA framework's reference architecture. They also shared the analysis of the operational requirements while designing their online Wintermute framework for HPC systems, which we extended during our requirements analysis. Their suggested high-level overview of the online ODA framework architecture, which could be integrated into a monitoring system, is shown in Fig. 2.7 [31]. The idea is to implement the ODA framework within an existing monitoring system, where the integration with monitoring daemons in compute nodes facilitates in-band operation and management nodes enable out-of-band operation, where the latter allows its interaction with a monitoring data broker, thus enabling access to streamed cluster-wide data as well as remote persistent storage.

Wintermute Architecture: Wintermute has been architected in a modular way driven by the requirements analysis, and is based on the analysis capabilities being supplied by *operator plugins* that are used to instantiate *operators*. Operators are the computational entities that perform all ODA tasks asynchronously, utilising a flexible local thread pool. These operators work on a set of *blocks*, which are container data structures that represent physical components like compute nodes or racks, or logical entities like user jobs in an HPC system). Each block has a set of *input sensors* which are used for the analysis and a set of *output sensors* which are responsible for storing the results of ODA operation, that are further either consumed by the monitoring system or some other operators (output sensors). A sensor captures some system information (whose reading has a numerical value and a time-stamp) and is defined to be an atomic monitoring entity (e.g., power, temperature, CPU counter, or ODA output). The architecture of Wintermute is presented in Fig. 2.8 which shows its interaction with the external components (excluding its integration into a monitoring system) [31]. The core components can be briefly summarised as follows:

- Operator Manager: It acts as the main interface between Wintermute and the monitoring system, and is responsible for loading requested operator plugins. It acts as a front-end interface responsible for exposing the available actions within the framework (e.g., start, stop or load plugins dynamically, trigger specific actions as signalled by the respective plugin).

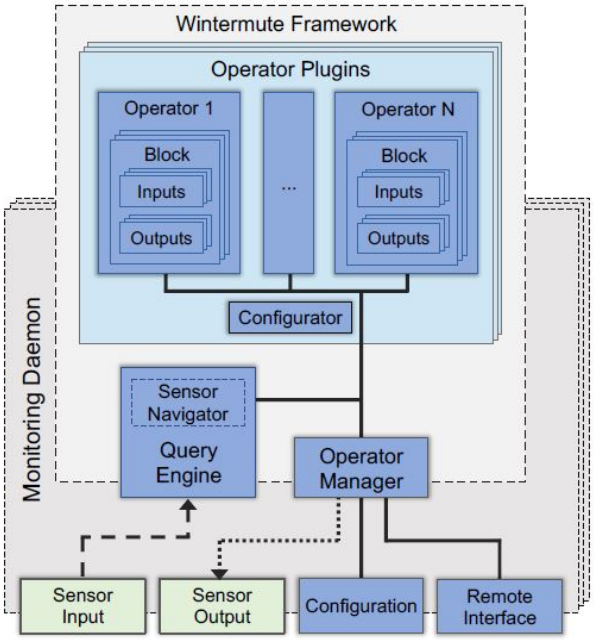


Figure 2.8: Architecture of Wintermute (abstract-level) (Source: [31]).

- Query Engine: It is a singleton component that exposes access to a *sensor navigator* object (which uses the block system to maintain a tree-like representation of the current sensor space) to operator plugins. Its uniform interface allows querying based on sensor name and time-stamp ranges.
- Operator Plugins: They are responsible for performing analysis by implementing a specific logic on the input sensor data alone. Job operator plugins are an extension that uses job-related data to produce output pertaining to a specific job. Plugins have two main components:
  - Operator: These objects perform the required analysis tasks whenever a computation is invoked for that operator.
  - Configurator: These components read plugins' configuration files, and instantiate the operators together with their blocks.
- Configuration: It is responsible for the initialisation of Wintermute, and grants access to the designated configuration files.
- Remote Interface: This is the interface exposed by the monitoring daemon which is used by Wintermute in turn to expose its data retrieval and remote control features.

Integration with DCDB Monitoring system: DCDB is a holistic monitoring framework for HPC systems which comprises several components enabling it to achieve a distributed and scalable architecture [32]. Wintermute is implemented in C++11 and is tightly coupled with the DCDB monitoring framework, as shown in the high-level overview of DCDB in Fig. 2.9 [31]. *Pushers* use a plugin-based architecture for the sampling of sensors on monitored components. *Collect Agents* collect the data sent to them via the MQTT protocol [33] and forward them to a *storage backend* (implemented using Apache Cassandra). Also, DCDB exposes a

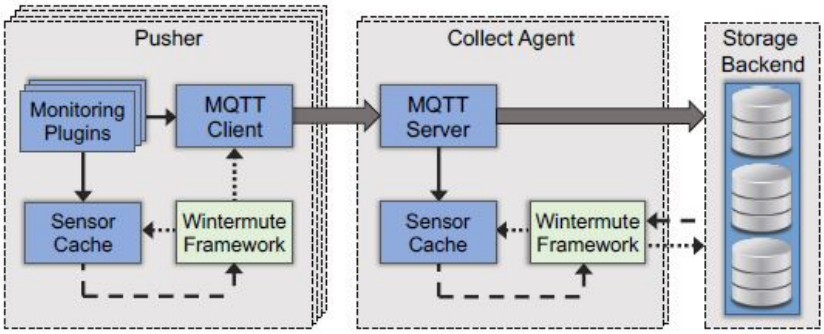


Figure 2.9: High-level overview of the architecture of DCDB, highlighting the Wintermute framework's integration in components and the data flow (Source: [31]).

RESTful API providing control of each of its components as well as sensor caches for faster in-memory access. The workflow of components involved is briefly discussed below (in the order presented by the respective literature authors):

- Operator Location: Operators can be instantiated in both pushers and collect agents by loading the appropriate plugins. Collect Agent location is optimal for system or infrastructure-level analysis because access to the entire system's sensor space is available (or the data could even be queried from the storage backend). Whereas, the operators only have access to locally sampled sensors and their sensor cache date in the pusher locations, thus making it optimal for runtime models which require low latency, data liveness, and horizontal scalability.

- Operational Modes: Operators can be configured to work in both online mode (at regular time intervals, resulting in continuous analysis) as well as in on-demand mode (by explicitly invoking the RESTful API for a specific block at scheduled intervals)

- Block Management: The blocks of a single operator can either be arranged as *sequential* when all corresponding blocks share a common operator (and are processed sequentially to avoid race conditions) or *parallel* if a distinct operator is created for each block, thus allowing parallel computation to improve scalability.

- Analysis Pipelines: The same format is shared by the output data produced by online operators (identical to all other sensor data), which facilitates creating pipelines of other operators' input as output to another operator. This functionality is very helpful in splitting the computational load between multiple locations, and implementing feedback loops (via control operators at the end of a pipeline) in an HPC system.

3. AI-driven Energy Aware Scheduling (EAS), IBM Research, STFC Daresbury Laboratory, UK

The IBM research team based in STFC Daresbury Laboratory, UK published a literature presenting an AI-driven holistic approach to energy and power management in data centres, which they describe as Energy Aware Scheduling (EAS) [34]. The aim of EAS is to look at energy efficiency across both hardware and software stacks of the IT infrastructure, including data centre(s), servers, network, cooling, IoT and Edge

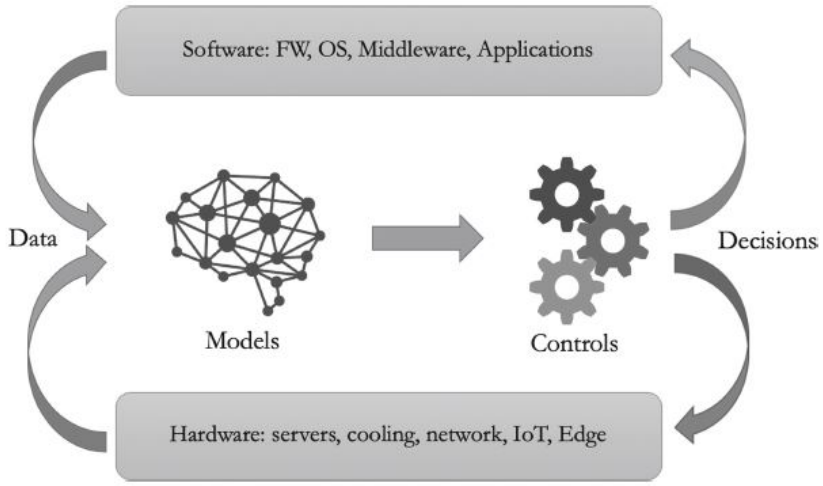


Figure 2.10: EAS concept: data continuously collected across hardware and software stacks and fed to the AI-based model for performance and power consumption predictions, which are then used by scheduling components to send control decisions back to the respective stacks (Source: [34]).

devices, to software stack ranging from firmware through to the OS, applications and workload managers. To achieve the same, EAS uses ML and AI methods for performance and power consumption modelling, and software-hardware co-design for implementing various energy/power-aware scheduling policies at different levels of the infrastructure. Fig. 2.10 broadly represents the main components of the Holistic EAS, where data is collected from a broad range of hardware devices and software components and fed into an AI component which generates some optimal decisions to meet required criteria, which are then sent back to respective hardware and software system components to re-implement [34].

Data Collection Framework: Fig. 2.11 represents the architecture used for data collection, where each component is housed in a Docker container resulting in numerous benefits like easier rolling upgrades independently from other components in the system, easy swapping of databases or graphing components, and easier portability to a different set of nodes (not necessarily IBM POWER servers), if required [34]. We briefly discuss the core components of the Data Collection Framework below:

- **Monitoring Tools**: The requirement for selecting a monitoring tool is to have as minimal impact on the cluster as possible, and thus clients running on the nodes are avoided for this reason. Amester tool is used to gather data from the Baseboard Management Controller (BMC) port on IBM POWER systems. BMC is a specialised service processor responsible for monitoring the physical state of the running system, and the BMC port connects to it, which helps Amester to gather out-of-band results as it will have zero impact on the running of the system and its resources. Another monitoring tool used is IBM Spectrum LSF Explorer because it integrates easily with the IBM Spectrum LSF, which is used as the workload manager. The authors are convinced that such a monitoring system could easily scale for exascale size and allow easier

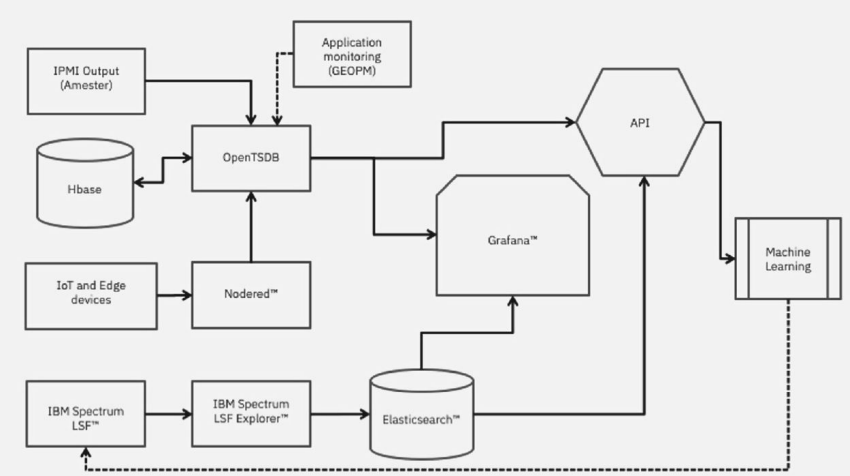


Figure 2.11: Overview of the intelligent data collection framework architecture (Source: [34]).

upgrade of the current system.

- **Metrics**: The requirement for metrics was driven by the need for them to be useful for general monitoring and ML, and be available on as many systems as possible. Hardware metrics considered for collection are fan power, CPU power and GPU power, and they are recorded every 250ms (the fastest the servers could read from the sensors). IBM Spectrum LSF Explorer is used for the collection of workload information which collects over 700 metrics every 10 seconds. Since this is an early analysis phase for the team, they are collecting all metrics and they can eventually identify the most useful ones for monitoring and feeding them into the ML systems. IoT-related metrics are sampled at an average rate of 5 seconds.

- **Databases**: Two separate databases are used given the nature of the system, to support Grafna. Elasticsearch is used to collect data from LSF, as it comes packaged with LSF Explorer and is already optimised to work with the high-frequency data sent from the former. The second database chosen is OpenTSDB, which is built on top of HBase.

- **Visualisation and API**: The visualisation is mainly centred around Grafana, which is very useful for generating graphs and dashboards to visualise the initial data and also allows the creation of live rotating dashboards. The data can also be accessed via API as the developers created a Python library (due to them using other ML libraries like Keras, PyTorch, and TensorFlow, which are all Python-based).

These frameworks are complex yet simplistic in understanding the design of an ODA functionality within a large-scale computing infrastructure, and provide a good explanation of the essential components required in an ODA framework. These frameworks have laid the groundwork for various ODA frameworks in the scientific domain including our design of an ODA framework for the data centre simulator OpenDC.

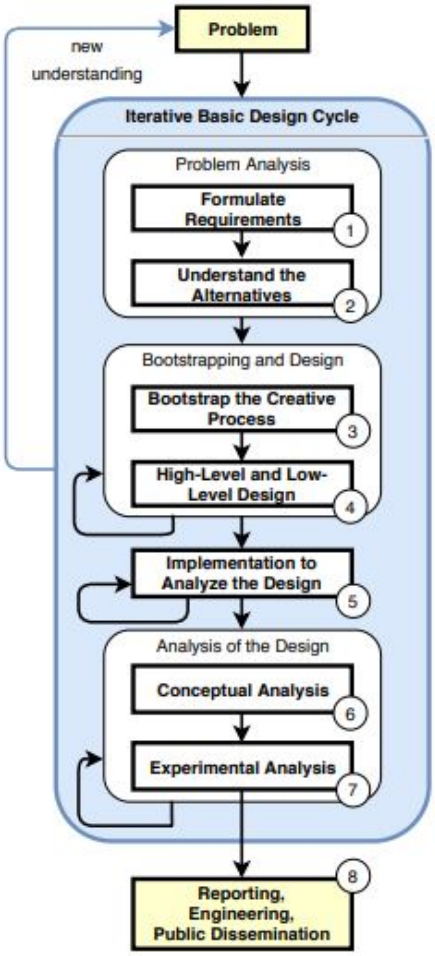


Figure 2.12: The @Large (or, AtLarge) design process (Source: [36]).

2.4 Ontology for HPC

An ontology encompasses a representation, formal naming, and definition of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains of discourse. The basic idea is to represent the properties of a subject area and their relationships, by defining a set of concepts and categories that represent the subject. One of the main reasons for designing an ontology for HPC is to make training datasets and AI models FAIR (FAIR data principles describe Findability, Accessibility, Interoperability, and Reusability) [35]. Some of the existing HPC ontology design already captures both high-level meta information as well as low-level data content for software, hardware, experiments, workflows, training datasets, AI models, and so on [6].

2.5 Design process of the @Large Research team

This project is supported under the affiliation of the "@Large Research" team, and supervised by the Professor who is the lead faculty and member of the @Large Re-

[6] `https://hpc-fair.github.io/ontology/`

search team. Thus, this project adheres to the principles mandated by the team, and aims to raise awareness amongst the scientific community (and general audience) of the operational data analytics (ODA) capabilities by showcasing the potential of energy-efficiency analysis of task-scheduling algorithms executing various types of workloads as part of data centre operations, and also highlight any node-anomaly resulting in higher power consumption at corresponding nodes. The mission of the @Large Research team (which is quoted below) supports designing and prototyping such critical systems which revolve around the critical topics related to energy-efficiency and sustainability concerns, including this project.

"*Our group is Massivizing Computer Systems (MCS), that is, making computer systems scalable, reliable, performant, etc., yet forming efficient, engineered ecosystems available to everyone.*
Our group aims to achieve this by conducting groundbreaking, impactful research, by educating the new generation of top-quality and socially responsible individuals, and by making innovation available to industry and society at large to improve the lives of millions."

The @Large design process is shown in Fig. 2.12, and the key principles from the vision literature authored by the team are listed below: [36]

- **P1**: Good design processes foster good system designs.
- **P2**: This is the Age of Distributed Ecosystems.
- **P3**: Dynamic non-functional properties and phenomena are first-class concerns.
- **P4**: Resource Management and Scheduling, and its interplay with various sources of information to achieve local and global Self-Awareness, are key concerns.
- **P5**: Education practices for MCS must ensure the competence and integrity needed for experimenting, creating, and operating ecosystems.
- **P6**: Design communities can foster and curate pragmatic, innovative, and ethical design practices.
- **P7**: We understand and create together a science, practice, and culture of MCS design.
- **P8**: We are aware of the history and evolution of MCS designs, key debates, and evolving patterns.

In a nutshell, the design principles takeaway that we have used in this thesis project are namely: focusing on the design process itself, brainstorming at the functional and non-functional requirements, pragmatic and innovative design adoption, and ensuring that the design is enabled with experimental evaluation, thus contributing to the science, practice, and culture of massivising computer systems (MCS).

2.6 Related work

This thesis project is split into two inter-related subprojects: the modelling of ontology for HPC systems based on publicly available cluster metrics, which paves the way for utilising the designed ontology to design an ODA framework for a data centre simulator (which relies on exact semantics from the modelled ontology) for analysis of the data centre operations. In this section, we go through some of the work which has been done individually under both sub-projects.

HPC ontology modelling

HPC ontology modelling work has already been done in previous scientific research. There are several research works already done in the field of HPC ontology, for example, by C. Liao et al. in [35], and by Castañé in [37], amongst others. One of the comprehensive HPC ontology designs that already captures both high-level meta information as well as low-level data content for software, hardware, experiments, workflows, training datasets, AI models, and so on is available as HPC Ontology[7]. There are other works of literature on HPC resources' ontology models like Zhou et al in [38], Zhao et al in [39], Tenschert A. in [40] and others, but those are simplified where the main goal of the authors has been to decompose applications between compute and data processes for HPC environments. There are several other works of literature presenting a unified ontology of cloud computing like Youseff et al in [41] and Imam, F.T. in [42]. Amongst these works, we did not find any literature studying the modelling of ontology derived from the metrics collected from operational HPC clusters. But, these metrics are the source of ODA framework design, and, so, the ODA framework should be driven by the modelled ontology, which should be derived from those captured metrics of large-scale computing infrastructures.

ODA capabilities implemented in data centre simulators

We explored various data centre simulators and found previous work based on scheduling algorithms comparison, but we found that they involved conducting experiments dedicated to only specific metrics. Thus, there is a lack of holistic ODA framework implementation for the data centre simulators, which could act as its digital twin presenting a virtual model for the same (reasons are presented already in Section 2.2).

We propose to use OpenDC [11] for simulating the execution of workloads within a data centre environment. However, the current version of OpenDC has limited visibility of the various metrics required as part of enabling ODA in a data centre environment, and it is also limited in the way it supports executing the workloads to reason about the events at deeper levels. Scientific works have been done by various authors using OpenDC, who have mainly focused on optimising various metrics like TDP (thermal design power). A holistic research work has not yet been performed under the umbrella term "ODA", which should focus jointly on the key pillars of an energy-efficient HPC or similar large-scale infrastructure, based on the definition shown in Fig. 2.3 [].
It must be noted that the thesis proposes to contribute to research on the convergence of scientific workloads in a large-scale computing environment, without

[7] `https://hpc-fair.github.io/ontology/`

generalising it to a specific environment with the aim that the ideas might be borrowed in other domains/environments. Some of the significant novel works done related to the field of ODA for HPC systems, which appear to be highly relevant to this thesis are listed below:

- **DCDB Wintermute: Enabling Online and Holistic Operational Data Analytics on HPC Systems** [31]: This appears to be the most extensively documented ODA framework "Wintermute", a novel generic framework to enable online ODA on large-scale HPC installations. The authors claimed to have implemented Wintermute on top of the holistic DCDB monitoring system, offering a large variety of configuration options to accommodate the varying requirements of ODA applications.

- **Collecting, Monitoring, and Analyzing Facility and Systems Data at the National Energy Research Scientific Computing Center** [29]: The authors describe their experiences in designing and implementing an infrastructure for extreme-scale operational data collection, known as the Operations Monitoring and Notification Infrastructure (OMNI) at the National Energy Research Scientific Computing (NERSC) centre at Lawrence Berkeley National Laboratory.

- **Revealing Power, Energy and Thermal Dynamics of a 200PF Pre-Exascale Supercomputer** [43]: This paper claims that they are the first to present the systematic analysis of power data of an HPC system at this scale.

- **A Study of Operational Impact on Power Usage Effectiveness using Facility Metrics and Server Operation Logs in the K Computer** [44]: This paper analyses the effect of operational impact on PUE (power usage effectiveness) using the facility metrics as well as the server operation metrics, and reveal that some maintenance operations degrade PUE.

- **Anomaly Detection and Anticipation in High Performance Computing Systems** [45]: The authors conduct an experimental evaluation on a tier-0 production supercomputer hosted at CINECA, Bologna, Italy, using the data/labels extracted from a service monitoring tool (Nagios) used by HPC system administrators to flag the nodes which undergo maintenance operations.

Chapter 3

Ontology modelling to drive the ODA design

In this chapter, we explore the modelling of ontology from the metrics collected from two HPC data centres, and how could they be useful for the purpose of Operational Data Analytics (ODA). The metrics collected are time-series data containing a sequence of observations ordered in time (a common method of monitoring and analysing the states of different components in a large-scale computing infrastructure).

3.1 Overview

Ontology is the field of science which helps us investigate what types of entities (or classes, sometimes also called concepts) exist in a domain of discourse, how they are grouped into categories, and how they are related to one another on the most fundamental level. An ontology along with a set of individual instances of classes constitutes a **knowledge base** [46]. Some of the common reasons for developing an ontology are briefly discussed below [46].

- Sharing a common understanding of the structure of information among the utilising resources (people or software agents): This is one of the most common goals in developing ontologies, which allows the ontology utilising resources to extract and aggregate information to answer user queries or as input data to other applications.
- Enabling reusability of domain knowledge: This is one of the driving forces behind the surge in ontology research. If we want to build a large ontology, several existing ontologies pertaining to various portions of the large domain can be integrated.
- Making explicit domain assumptions: This allows the assumptions underlying an implementation to be changed later if the knowledge about the domain changes. It also allows newer users to be familiar with the explicit specifications of the domain knowledge.
- Analysing domain knowledge: Once a declarative specification of the terms is available, the domain knowledge can be analysed formally accordingly.

Developing an ontology includes the following in practical terms (which is the basis for our ontology modelling):

Object-oriented programming	Formal ontology
Object	Entity
Module	Ontology
Class	Class
Class inheritance	Class inheritance, also called "is-a" relation
— (no equivalent)	Property inheritance
Instance	Individual
Attribute or property	Property, role, or predicate
Value of an attribute for an instance	Relation
Class name	IRI
Datatype	Datatype
Method	— (no equivalent)
— (no equivalent)	Logical constructor
	Restriction
	Disjoint

Figure 3.1: A correspondence between the concepts of object-oriented programming and formal ontologies (Source: [47]).

• defining classes (sometimes called concepts) in the ontology,
• arranging the classes in a taxonomic hierarchy (subclass-superclass order),
• defining various features and attributes of the concepts (also called slots, or roles of properties), and
• filling in the values for slots for instances of classes.

OWL (Web Ontology Language)

The Web Ontology Language (OWL) is a formal language for expressing ontologies and is based on the description logic (DL). The underlying format which is fundamental to storing a wide range of information in OWL-based ontologies is the Resource Description Framework (RDF). The information stored in an RDF consists of a triple: subject, property, and object. For instance, ("DataCentre-XYZ", hasAmbientTemperature, "21 degrees") states that a DataCentre-XYZ has an ambient temperature of 21 degrees (which could further be explicitly related to the time of record capture using another property, or annotated with the timestamp information). It must be noted that an ontology can be queried using a standard RDF query language named SPARQL, short for "SPARQL Protocol And RDF Query Language".

In this thesis project, OWL has been used for modelling the ontology (where OWL support is made available in the Python platform by Owlready2) [48], and a few experiments (using the SPARQL language) have been conducted to validate that the designed ontology meets the expectations by satisfying the requirements outlined in the next section. Some of the key definitions which set up the basis of the ontology modelling using OWL are outlined below:

Classes: Entities in an object-oriented world
Inheritance: Inheritance is called "is-a" relationship
Disjoints: Two classes cannot have individuals in common (a given grouping can-

not be both x and y)
Partitions: class a and b constitute a partition of class AB (when AB is always either a or b)
Data properties: properties whose values are data (number, text, date, boolean, etc.)
Object properties: properties whose values are entities (ontology individuals)
Annotation Properties: properties that can mix data and entities without restriction

• OWL supports inheritance between properties, in addition to inheritance between classes

• Each "data property" can be configured by specifying: the domain (the class for which the property is defined), range (associated datatype), and its functional status (a given individual can have only one value for this property if functional, or if it can have several values)

• Each "object property" can be configured by specifying: the domain (the class for which the property is defined), range (associated objects), its inverse property (existing relationships when the property is read backwards), its functional status (a given individual can have only one value for this property if functional, or if it can have several values), its inverse functional status (if the inverse property is functional), its transitive status (if it is possible to chain the property on several objects), its symmetric status (if it can be read indifferently in both directions), its asymmetric status (if it is never symmetrical), its reflexive status (if it always applies between any object and itself), and its irreflexive status (if it is never reflexive)

• We can also add restrictions to the classes: "existential restriction" or *some* (the class of individuals who have at least one relation of a certain property with an individual belonging to a certain class), "cardinality restrictions" or *exactly/min/max* (the class of individuals who have either exact or a minimum or a maximum number of relations of a certain property with an individual belonging to a certain class), "universal restriction" or *only* (the class of individuals who only have a relation of a certain property with one or more individuals belonging to a certain class), and "value restriction" or *value* (the class of individuals who have a certain value for a certain property; sometimes called role-filler)

• OWL allows the use of logical operators as constructors: logical AND or intersection (individuals belonging to several classes at the same time), logical OR or union (individuals belonging to a class among several), and logical NOT or compliment (individuals who do not belong to a given class)

• OWL also allows using a formal equivalence definition via an equivalence relation (the defined classes allow reclassifying individuals during automatic reasoning).

3.2 Requirements analysis

We need to understand how ontology is modelled. Before doing that, we first need to understand the relationship or correspondence between the object-oriented field of programming and the field of ontology. The table shown in Fig. 3.1 represents the correspondence between the two fields [47].

Functional requirements

Here we discuss some of the functional requirements governing our ontology model:
• **Time-series modelling**: The ontology should support the modelling of attributes extracted from time-series metrics collected from a large-scale computing infrastructure, e.g., an HPC cluster.
• **Resource description**: The ontology should describe the HPC cluster resources in a structured way, including details about nodes, processors, etc.
• **Performace metrics**: The ontology should capture and analyse performance metrics such as resource utilisation and energy consumption.
• **Consistent and Accurate**: The ontology should be consistent and accurate in its representation by reflecting the state of the HPC cluster and its resources.

Non-functional requirements

Below, we discuss some of the non-functional requirements governing our ontology model.
• **Interoperability**: The ontology should be designed with interoperability in mind, which should facilitate integration or reuse with/by other ontologies.
• **Usability**: The ontology should be user-friendly in nature, having sufficient comments or labels for accessibility by both experts and non-experts.
• The ontology might be based on the international, European or de facto standards in existence.
The best resource which is closer to our ontology requirements is a work of literature studying the ontological representation of time-series observations on the Semantic Sensor Web [49]. It suggests the usage of three important modelling classes, out of which we find that the most relevant class "Observation" can be reused, and thus discussed in detail below. The two other classes "ObservationCollection" and "TimeSeriesObservation" do not seem convincingly reusable, mainly because of the nature of the requirements to model a sample time-series data. If we were to model multiple time-series data in the ontology, then we could have inherited the same structure. But, in this project, we limit ourselves to demonstrating a single set of records in the ontology model, other than the established relations amongst the classes.

• Observation class (an act of observing a property or phenomenon with the intent to produce an estimate of the value of the property). Some of the relationships for observations which have been used here are listed below:
featureOfInterest (representation of the object being observed)
observedProperty (the phenomenon for which the observation result provides an estimate of its value)
samplingTime/**generatedAtTime** (the time when the phenomenon was measured)

result/**value** (an estimate of the value of a property generated using a known procedure)
memberOf (a relation to a set of observations of observation collection)

3.3 HPC ontology modelling

Class	Parent-class
Hardware	Thing
Computer	Hardware
Cluster	Computer
Rack	Hardware
Server	Computer
Processor	Hardware
CPU	Processor
Coprocessor	Processor
Accelerator	Coprocessor
GPU	Accelerator
Memory	Hardware
Concept	Thing
Entity	Concept
Site	Entity

Class	Parent-class
Artifact	Thing
Data	Artifact
Software	Entity
MonitoringSystem	Software
MetricsExporter	Software
MonitoringMetrics	Data
ResourceManager	Software
JobScheduler	Software
Job	Data
Resource	Entity
Feature	Thing
Property	MonitoringMetrics
MetricCollector	Feature

Table 3.1: List of classes that form the basic hierarchy which is commonly used in both SURF's & CINECA's cluster-exported metrics ontology modelling.

As the basics have been discussed sufficiently, we proceed with the steps taken for modelling ontology based on metrics collected from 2 different HPC clusters running different data exporters (SURF's Prometheus vs CINECA's IPMI). We highlight the base classes required for ontology modelling, which have been referenced from the existing popular HPC ontology model [35]. The base classes in an ontology are all derived from the special 'Thing' class. This representation is listed below in the form of child-class(parent-class), and can be read as a child-class inherits from parent-class.

• The core classes which have been inherited (we tried to reuse the classes already modelled in HPC as much as possible) are listed in Table 3.1.

• The entries in the table must be read from the left column from top to bottom, and then to the right column from top to bottom. Each of the row entries can be interpreted as the leftmost entry being a child class inheriting from the parent class entry appearing on the right side.

Further, properties which are distinct from SURF's metrics or CINECA's metrics are discussed later in their respective sections.

SURF's LISA cluster metrics (relational) description

• The key classes (or, concepts) and properties of HPC ontology involving SURF's LISA cluster metrics are described in Table 3.2 and 3.3 respectively.

Class	Parent-class
DataCentre	Site
AmbientTemperature	Property
PowerUsage	Property
DeviceStatus	Property
TimeSinceEpoch	Property
Motherboard	Hardware
ThermalZone	Feature
Temperature	Property
SystemStatisticsCollector	MetricCollector
ProcessesBlocked	Property
ProcessesRunning	Property
MemoryStatisticsCollector	MetricCollector
ActiveBytes	Property
DirtyBytes	Property
MemFreeBytes	Property
PerCPUBytes	Property
DiskIOStatisticsCollector	MetricCollector
NumberOfIOs	Property
TotalWrites	Property
ReadBytes	Property
WrittenBytes	Property
HardwareMonitor	MetricCollector
NodeTemperature	Property
CPULoadCollector	MetricCollector
CPULoadAverage	Property
ARPMetricCollector	MetricCollector
ARPEntries	Property
BootTimeMetricCollector	MetricCollector
BootTime	Property
EnergyUsageMetricCollector	MetricCollector
RunningAveragePowerLimit	Property
UDPQueuesCollector	MetricCollector
UDPQueues	Property
VirtualMemoryStatisticsCollector	MetricCollector
ContextSwitches	Property
Forks	Property
Interrupts	Property
FileSystemMetricCollector	MetricCollector
FilesFree	Property
Files	Property
FreeSpace	Property
DeviceErrorReport	Property
FileSystemSpace	Property
FileSystemSize	Property
GPUDevice	Feature
FanSpeed	Property
Temperature	Property
PowerUsage	Property
DutyCycle	Property
Memory	Property
NetworkInterface	Feature
NetworkInterfaceMetrics	Property
KernelNetworkSubsystem	Feature
ICMPMetrics	Property
TCPMetrics	Property
UDPMetrics	Property
ExitCode	Data

Table 3.2: List of classes that form the hierarchy which is used in SURF's cluster-exported metrics' ontology modelling.

• It must be noted that aggregation operators such as sum, min, max and avg have been used in several metrics in the dataset, before exporting these metrics to one of the available Prometheus Exporters (Splunk).

Property	Property-type	Nature of value (Domain » Range)	Description
hasMember	Object property	Thing » Thing	shows membership relation between two entities
featureOfInterest	Object property	Thing » Feature	representation of the object being observed
manageJob	Object property	Software » Job	software manages jobs

hasMonitoringSystem	Object property	Computer » MonitoringSystem	computer has monitoring system
hasMetricsExporter	Object property	Entity » MetricsExporter	an entity has metrics exporter
exportMonitoringMetrics	Object property	Software » Job	software manages jobs
observedProperty	Object property	Thing » Property	the phenomenon for which the observation result provides an estimate of its value
generatedAtTime	Data property	Thing » datetime	timestamp of event generation
hasMeasurementIdentifier	Data property	MonitoringMetrics » int	metrics has identifier
measuresValueOfThing	Object property	MonitoringMetrics » Thing	metrics measures value of a thing
hasTemperatureValue	Data property	Feature » float	temperature value of the feature
hasPowerUsageValue	Data property	Property » float	power-usage value of the property
atLocation	Object property	Thing » Thing	thing is at some location
hasUpValue	Data property	Property » float	system up value
hasTimeSinceEpochValue	Data property	Property » datetime	property has epoch timestamp
hasNumberOfProcessesBlockedValue	Data property	Property » float	blocked processes count
hasNumberOfProcessesRunningValue	Data property	Property » float	running processes count
hasActiveBytesValue	Data property	Property » float	memory field Active_bytes
hasDirtyBytesValue	Data property	Property » float	memory field Dirty_bytes
hasMemFreeBytesValue	Data property	Property » float	memory field MemFree_bytes
hasPerCPUBytesValue	Data property	Property » float	memory field Percpu_bytes
hasNumberOfIOsInProgressValue	Data property	Property » float	number of I/Os in progress
hasTotalNumberOfWritesCompletedValue	Data property	Property » float	number of writes completed
hasTotalNumberOfReadBytesValue	Data property	Property » float	number of bytes read
hasTotalNumberOfWrittenBytesValue	Data property	Property » float	number of bytes written
hasNodeTemperatureValue	Data property	Property » float	temperature of the node
hasCPULoadAverageValue	Data property	Property » float	CPU load average
hasArpEntriesValue	Data property	Property » float	ARP entries by device
hasBootTimeValue	Data property	Property » float	node boot time

hasRAPLPackageValue	Data property	Property » float	Running Average Power Limit package value
hasAllocatedMemoryForUDPDatagramsValue	Data property	Property » float	memory allocated in the kernel for UDP
hasNodeTemperatureValue	Data property	Property » float	temperature of the node
hasTotalContextSwitchesValue	Data property	Property » float	total context switches
hasTotalForksValue	Data property	Property » float	total number of forks
hasTotalInterruptsServicedValue	Data property	Property » float	number of interrupts serviced
hasTotalFreeFileNodesValue	Data property	Property » float	total free file nodes (filesystem)
hasTotalFileNodesValue	Data property	Property » float	total file nodes (filesystem)
hasTotalFreeSpaceValue	Data property	Property » float	free space (filesystem)
hasDeviceErrorValue	Data property	Property » float	total error occurred while getting device statistics
hasAvailableFileSystemSpaceToNonRootUsersValue	Data property	Property » float	space available to non-root users (filesystem)
hasFileSystemSizeValue	Data property	Property » float	size (filesystem)
hasFanSpeedValue	Data property	Property » string	fanspeed as a percent of its max (GPU device)
hasDutyCycleValue	Data property	Property » string	percent of time of kernel execution on the GPU device
hasMemoryUsedValue	Data property	Property » float	memory used (GPU device)
hasReceiveBytesTotal	Data property	Property » float	network device statistic receive_bytes
hasReceivePacketsTotal	Data property	Property » float	network device statistic receive_packets
hasReceiveMulticastPacketsTotal	Data property	Property » float	network device statistic receive_multicast
hasReceiveDroppedPacketsTotal	Data property	Property » float	network device statistic receive_drop
hasTransmitBytesTotal	Data property	Property » float	network device statistic transmit_bytes
hasTransmitPacketsTotal	Data property	Property » float	network device statistic transmit_packets
hasICMPInErrorsMessagesValue	Data property	Property » float	statistic IcmpInErrors
hasICMPInMessagesValue	Data property	Property » float	statistic IcmpInMsgs

hasICMPOutMessagesValue	Data property	Property » float	statistic IcmpOutMsgs
hasTCPInErrorsSegmentsValue	Data property	Property » float	statistic TcpInErrs
hasTCPInSegmentsValue	Data property	Property » float	statistic InSegs
hasTCPOutSegmentsValue	Data property	Property » float	statistic TcpOutSegs
hasTCPRetransmittedSegmentsValue	Data property	Property » float	statistic TcpRetransSegs
hasUDPInDatagramsValue	Data property	Property » float	statistic UdpInDatagrams
hasUDPOutDatagramsValue	Data property	Property » float	statistic UdpOutDatagrams
hasUDPInErrorsDatagramsValue	Data property	Property » float	statistic UdpInErrors
hasResourceManager	Object property	Computer » ResourceManager	a computer has a resource manager
hasJobScheduler	Object property	Computer » JobScheduler	a computer has a job scheduler
managesJob	Object property	JobScheduler » Job	job scheduler manages job
hasJobIdentifier	Data property	Job » int	job has job-ID
hasStartDate	Data property	Thing » datetime	thing (job) has a start-date
hasEndDate	Data property	Thing » datetime	thing (job) has an end-date
isScheduledOn	Object property	Data » Thing	data (job) is scheduled on something (node)
hasNodeType	Data property	Server » string	type of node
usesNode	Data property	Job » int	number of nodes used for the job
usesCore	Data property	Job » int	number of cores used for the job
requiresSharedNode	Data property	Job » string	whether requires shared node
submittedAtTime	Data property	Job » datetime	submission time of job
startedAtTime	Data property	Job » datetime	start time of job
endedAtTime	Data property	Job » datetime	end time of job
hasState	Data property	Job » string	job end state
hasExitCode	Data property	Job » string	exit code of job
hasExitCodeValue	Data property	string » int	exit code value of job
hasReservationIdentifier	Data property	Job » string	reservation for the job

Table 3.3: High-level concepts of the HPC ontology involved in modelling SURF's LISA cluster metrics.

CINECA's Marconi100 cluster metrics (relational) description

Marconi 100 is the new accelerated cluster based on IBM Power9 architecture and Volta NVIDIA GPUs, acquired by Cineca within PPI4HPC European initiative[1]. It has been made public to the Italian public and industrial researchers since April 2020, and its computing capacity is about 32 PFlops.

• The dataset metric used in this project for ontology modelling is made available by the IPMI plugin, which collects the sensor data provided by the OOB management interface (BMC) of cluster nodes [50].

• The dataset used for ontology modelling in this thesis project is a subset of the huge dataset made publicly available by the Unibo[2] and CINECA computing centre[3] teams [51].

• The key classes (or, concepts) and properties of HPC ontology involving CINECA's Marconi100 cluster metrics are described in Table 3.4 and 3.5 respectively.

Class	Parent-class	Class	Parent-class
IPMIProperty	Property	DriveShelfFanModule	Feature
Temperature	IPMIProperty	Fan	Feature
AmbientTemperature	IPMIProperty	DiskFan	Feature
Speed	IPMIProperty	GPUCore	Feature
PowerUsage	IPMIProperty	GPUMemory	Feature
Voltage	IPMIProperty	GraphicsCard	Feature
Current	IPMIProperty	CPUSocket	Feature
NVIDIAQuadroGV100Card	IPMIProperty	CPUSocketCore	Feature
Accelerator	IPMIProperty	InputOutputSubsystem	Feature
Organization	Thing	MemorySubsystem	Feature
Company	Organization	VoltageRegulator	Feature
NVIDIA	Company	PCIExpressSlot	Feature
ServerInlet	Feature	PowerSupplyInput	Feature
DualInlineMemoryModule	Feature	PowerSupplyOutput	Feature

Table 3.4: List of classes used dedicatedly in CINECA's cluster-exported metrics' ontology modelling.

Modelling SURF's LISA cluster metrics

The metrics reflected in the modelled ontology for SURF's LISA cluster metrics are shared below in Table 3.6, which is quite extensive in modelling all classes and properties (including sample data consisting of one record for each of the classes, properties and their respective relationships) available in the shared dataset. We share the relevant snippet from LISA's metric-based ontology modelling, as the overall image size is too large to fit into this report. However, interested readers can access the same from the GitHub page, where the link of how to access and visualise the owl file is also discussed in sufficient detail in the Appendix at the end of this report.

[1]Marconi 100 - `https://www.hpc.cineca.it/hardware/marconi100`
[2]University of Bologna - `https://www.unibo.it/en`
[3]CINECA - `https://www.cineca.it/en`

Property	Property-type	Nature of value (Domain » Range)	Description
hasMember	Object property	Thing » Thing	shows membership relation between two entities
featureOfInterest	Object property	Thing » Feature	representation of the object being observed
isAssociatedWith	Object property	Organization » Thing	organisation is associated with
hasMonitoringSystem	Object property	Computer » MonitoringSystem	computer has monitoring system
hasMonitoringMetrics	Object property	Entity » Data	an entity has monitoring metrics
observedProperty	Object property	Thing » Property	the phenomenon for which the observation result provides an estimate of its value
generatedAtTime	Data property	Thing » datetime	timestamp of event generation
measuresValueOfThing	Object property	MonitoringMetrics » Thing	metrics measures value of a thing
hasTemperatureValue	Data property	Feature » float	temperature value of the feature
hasSpeedValue	Object property	Property » int	speed value of the property
hasPowerValue	Object property	Property » int	measured power value
hasValue	Data property	Property » int	possesses some value
hasVoltageValue	Data property	Property » int	property has voltage
hasCurrentValue	Data property	Property » int	property has current

Table 3.5: High-level concepts of the HPC ontology involved in modelling CINECA's Marconi100 cluster metrics.

Modelling CINECA's Marconi100 cluster metrics

The metrics reflected in the modelled ontology for CINECA's Marconi100 cluster metrics are shared below in Table 3.7, which successfully modelled all classes and properties (including sample data consisting of one record for each of the classes, properties and their respective relationships) available in the shared dataset. We share the relevant snippet from Marconi100's metric-based ontology modelling, as the overall image size is too large to fit into this report. However, interested readers can access the same from the GitHub page, where the link of how to access and visualise the owl file is also discussed in sufficient detail in the Appendix at the end of this report.

Axiom	710
Logical axiom count	442
Declaration axiom count	230
Class count	82
Object property count count	17
Data property count	63
Individual count count	69
Annotation property count	2

Table 3.6: Ontology metrics related to the modelled SURF's LISA cluster.

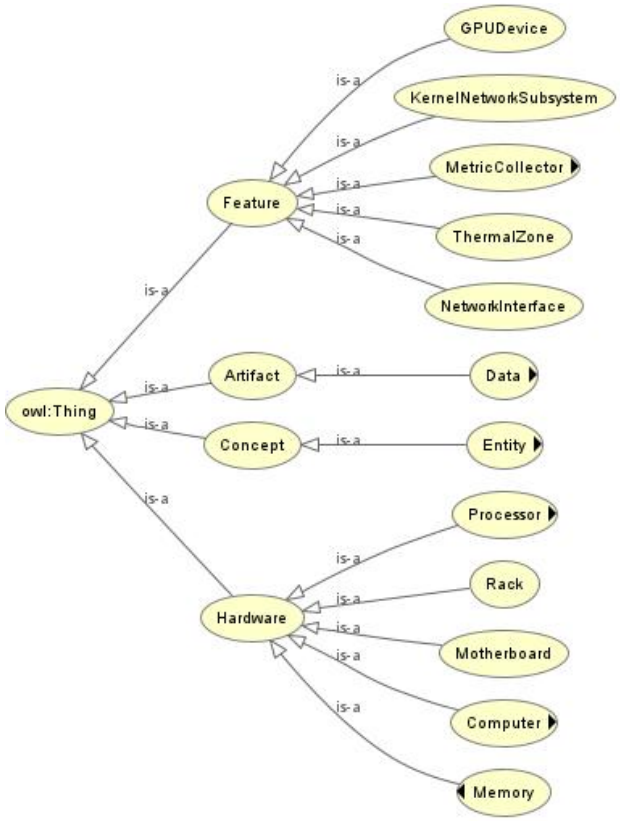


Figure 3.2: A class-hierarchy representation capturing the ontology for a dataset pertaining to SURF's LISA HPC cluster

Axiom	384
Logical axiom count	250
Declaration axiom count	98
Class count	47
Object property count count	10
Data property count	7
Individual count count	35
Annotation property count	2

Table 3.7: Ontology metrics related to the modelled CINECA's Marconi100 cluster.

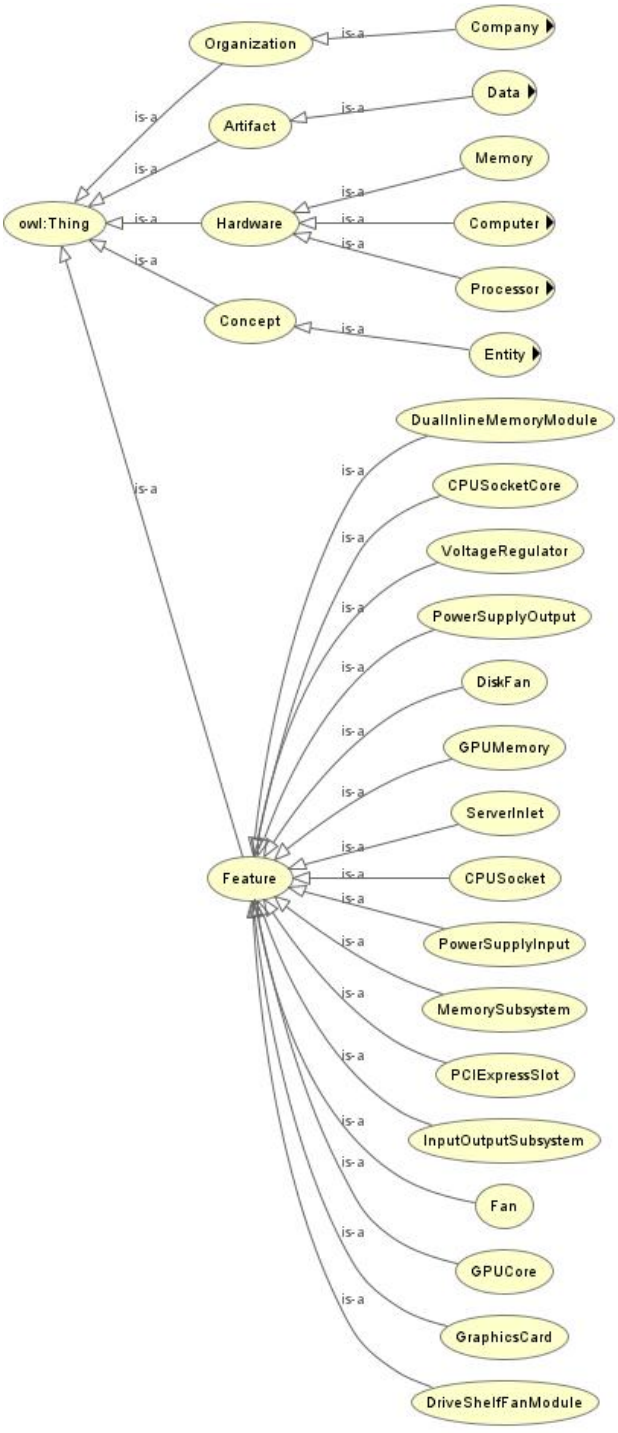


Figure 3.3: A class-hierarchy representation capturing the ontology for a dataset about CINECA's Marconi100 cluster

```
[[hpcontology_surf.Property],
 [hpcontology_surf.Memory],
 [hpcontology_surf.AmbientTemperature],
 [hpcontology_surf.PowerUsage],
 [hpcontology_surf.DeviceStatus],
 [hpcontology_surf.TimeSinceEpoch],
 [hpcontology_surf.Temperature],
```

(a) SURF's ontology SPARQL output

```
[[hpcontology_cineca.Property],
 [hpcontology_cineca.IPMIProperty],
 [hpcontology_cineca.Temperature],
 [hpcontology_cineca.Speed],
 [hpcontology_cineca.PowerUsage],
 [hpcontology_cineca.Voltage],
 [hpcontology_cineca.Current],
 [hpcontology_cineca.NVIDIAQuadroGV100Card],
 [hpcontology_cineca.AmbientTemperature]]
```

(b) CINECA's ontology SPARQL output

Figure 3.4: Modelled ontology's SPARQL output showing the ambient temperature property, amongst others

3.4 Experimental validation

The very first scenario is to validate that the modelled ontologies meet the requirements of an ontology in terms of structural modelling. It should be noted that the overall ontology structure could be validated to be syntactically correct using the graph dump property (ontology.graph.dump() in Python's 'Owlready2' module[4]), to ensure that there are no errors encountered while representing data with the ontology (e.g., datatype mismatch, incorrect attribute-name, data missing scenario). If the modelled ontology is structurally incorrect while adding data, then the graph dump statement should give an error.

- **Exp-1**: Validation of some of the key properties which are common in both ontologies (like ambient temperature, host power usage, etc.)
  → We perform a SPARQL query to validate that the results are matched in both ontologies, as shown below.

```
# Owlready2 stores every triple in a 'World' object, and it can handle several Worlds in parallel. 'default_world' is the World used by default.
result_surf = list(default_world.sparql("""
                   PREFIX  <https://example.org/hpcontology_surf.owl#>
                   SELECT DISTINCT ?x where{
                   ?x rdfs:subClassOf* Property .
                   }
                   """))
result_cineca = list(default_world.sparql("""
                   PREFIX hpc: <https://example.org/hpcontology_cineca.owl#>
                   SELECT DISTINCT ?x where{
                       ?x rdfs:subClassOf* hpc:Property .
                   }
                   """))

```

The output of both the SPARQL queries from SURF's and CINECA's ontology are shown in Fig. 3.4. We can see that the common properties like 'AmbientTemperature' and 'PowerUsage' are captured in the modelled ontology, which is successfully retrieved based on the SPARQL query. Additionally, this SPARQL lists all the properties which have been modelled in both ontologies based on the metrics shared in

[4]Owlready2 - https://owlready2.readthedocs.io/en/latest/

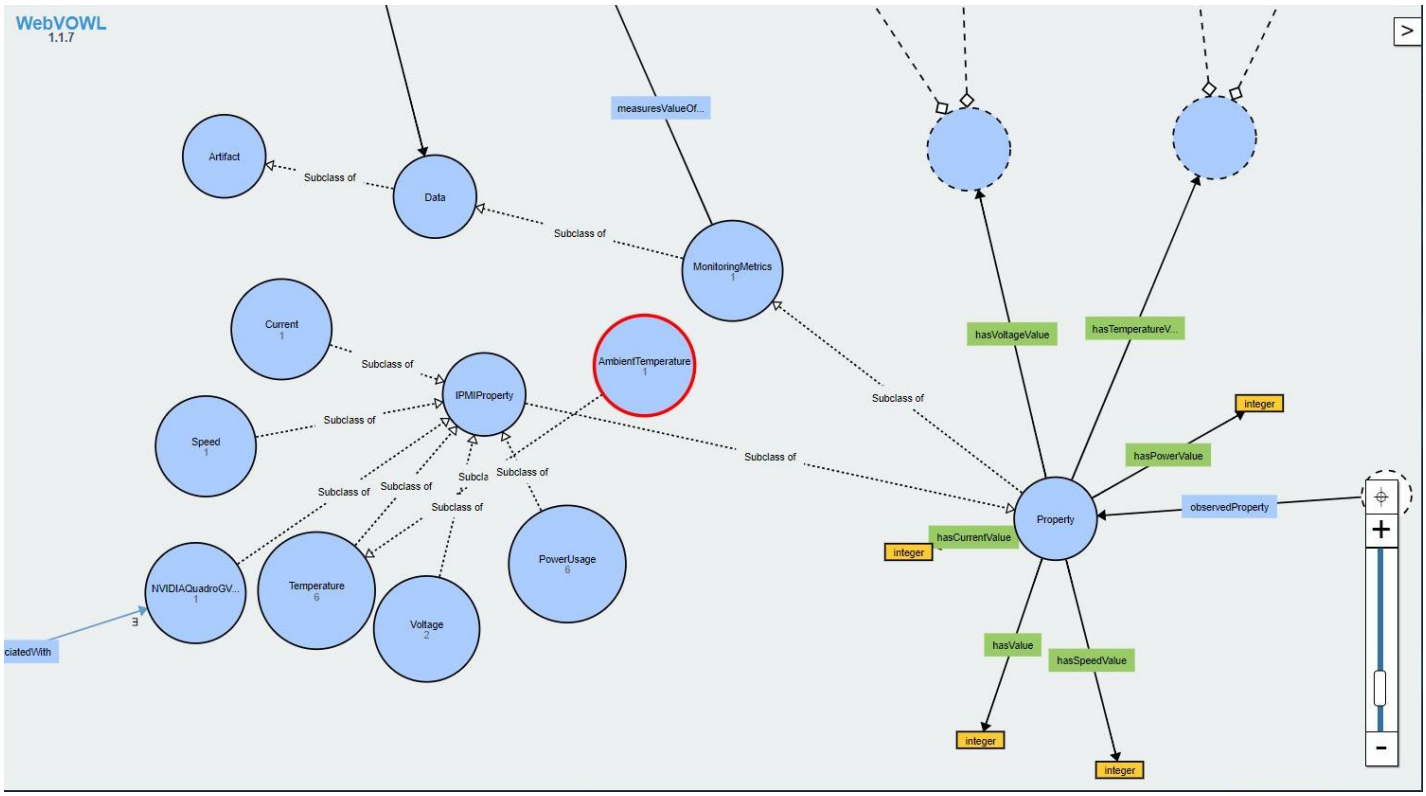


Figure 3.5: CINECA's OWL ontology visualisation (using WebVOWL) showing the ambient temperature property, amongst others

their datasets respectively.

- **Exp-2**: Visualisation to validate that the monitoring metrics relationship is preserved with the captured properties and includes key properties which can (as collected in the metrics)
  → The goal of this experiment is to verify that the ontology modelled for the available datasets satisfies the key relations which can be used for scientific work, such as the representation of data centre ambience temperature which is captured in their datasets, which can be used while researching coolant-related topics in data centre infrastructure.
  Here, we share the snippet from CINECA's Marconi 100 modelled ontology in Fig. 3.5 which shows that the essential properties like ambient temperature are captured. We use the WebVOWL [52] online visualisation of the OWL ontology file provided by TIB-EU[5] to verify that the visualisation reflects the key classes and properties in a relationship.

- **Exp-3**: Verification that the modelled ontology also reflects the graphical layout, which could be inferred to create general graph structures on the fly for analytical purposes (e.g., for performing ODA-related analysis)
  → The goal of this experiment is to verify that the ontology modelled for both HPC clusters can be visualised in the form of graphical layouts with nodes and edges, which could be easily converted to a graphical structure in a graph database using available tools.
  The resulting graphical layout for SURF's LISA cluster is visualised using Web-

[5]TIB-EU: `https://service.tib.eu/webvowl/`

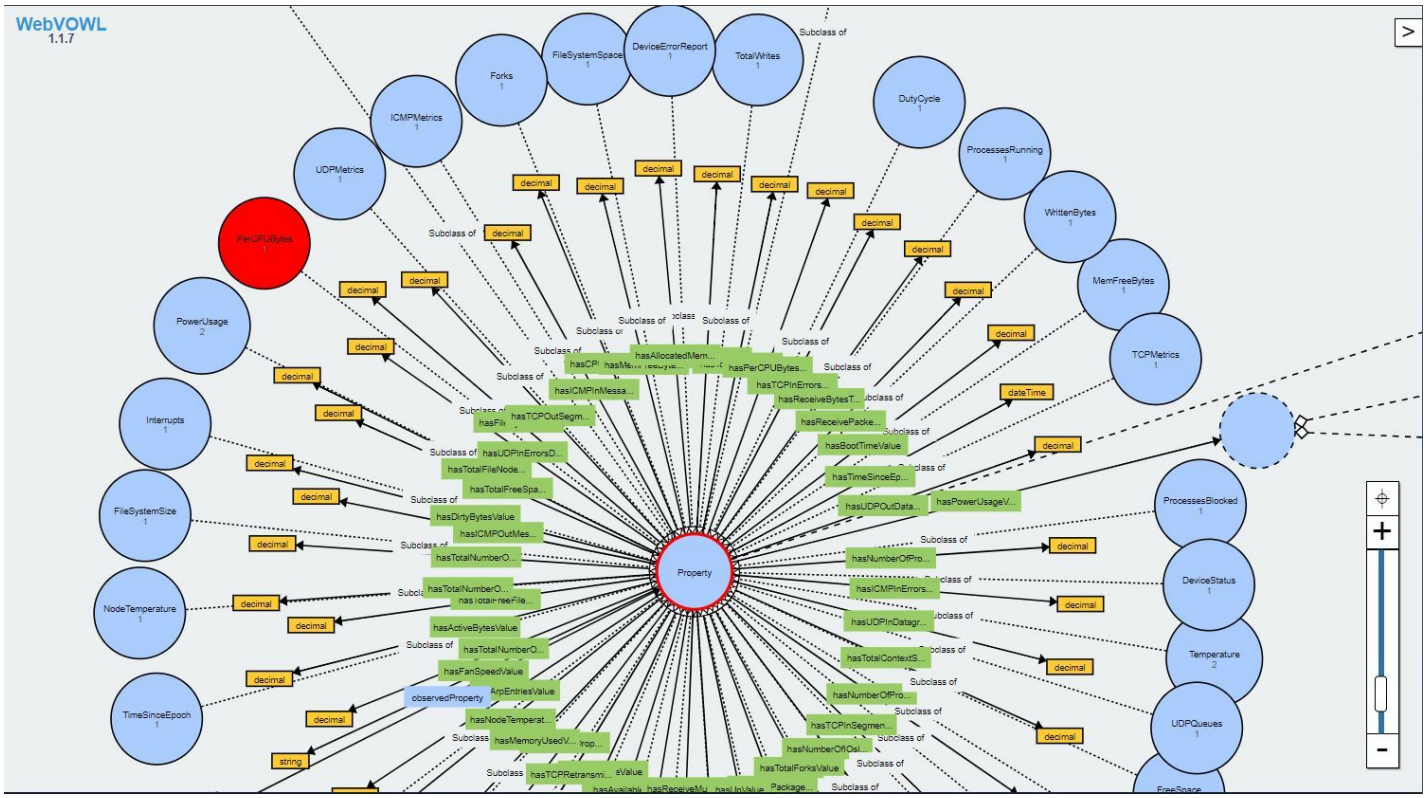


Figure 3.6: SURF's OWL ontology visualisation (using WebVOWL) showing the graph layout according to the VOWL specification, and listing graphical nodes and edges information on the right side information box

VOWL ontology visualisation and shown in Fig. 3.6.

There are tools like Onto2Graph[6] that enable the creation of general graph structures from the OWL ontologies using automated reasoning [53]. Once the graph structure is available in a graph database, graphical analysis can be performed as conducted in one of the works of literature based on climatic-data analysis [54].

Strength of this ontology project

An important point worth sharing is the collaboration with the Unibo team, Metaphacts team, and other stakeholders via Graph-Massivizer [55] and GraphGreenifier[56] projects, and this ontology modelling serves as a stepping stone highlighting how the modelled HPC ontology could be used by CINECA/Unibo teams by converting to graphical data format in a graph database, and further operational data analytics (like node anomaly detection) could be done. Additional work could be done to integrate other HPC components for more advanced analytical capabilities (not included in the current scope), which form future work in this field.

3.5 Summary

In this section, we discussed the significance of ontology, and how could this be useful for gaining insights about various metrics in the dataset. The ontology provides a nice

[6]Onto2graph - `https://github.com/bio-ontology-research-group/Onto2Graph`

relationship among various components of the large-scale computing infrastructure, which is difficult to study without proper modelling. This could help in the simplified correlation and analysis of various metrics whose relationships earlier were difficult to establish and correlate by the researchers. It is important to note that the 2 ontologies were modelled individually from the point of segregation of both clusters that are from the heterogeneous environments, however, both ontologies have a lot of common concepts (classes) and properties (promoting reusability and uniformity) of the HPC computing infrastructure.

To the best of our knowledge, this is the first attempt to explicitly address an ontology model supporting HPC metrics available in time-series format, and that too using multiple (two) datasets covering unique as well as different aspects of ontology (due to different host data collectors targeting different components of the infrastructure).

Chapter 4

Design of 'ODAbler': an ODA framework for a data centre simulator (OpenDC)

In this section, we present the design of the ODAbler framework to perform operational data analytics on a large-scale computing infrastructure configured within the OpenDC simulator, mimicking the real-world environment and operations in the simulator to a close extent.

4.1 Why is an ODA framework needed?

An operational data analysis framework is needed in production data centres for several important reasons: to enable energy efficiency of the infrastructure, proactive performance monitoring, capacity planning, fault detection and mitigation, security-related safeguarding, and resource optimisation to name a few. We have seen some of the popular ODA frameworks being used in production in Section 2. Netti et al. proposed a macroscopic-level classification of Operational Data Analytics depending on the purpose of the technique being used [57], namely Operational Data Analytics for Visualisation (ODAV) and Operational Data Analytics for Control (ODAC), as shown in Fig. 4.1. The key stages involved in a simplistic ODA pipeline are listed below:

- **Monitoring**: The monitoring of data centre infrastructure, hardware or software metrics is the basis of enabling an ODA pipeline.
- **Monitoring data processing**: The monitoring data generated for the data centre components needs to undergo data processing for further analysis.
- **Operational Data Analytics**: The analytics performed on the relevant monitoring metrics to gain some insights about the behaviour of the computing infrastructure at various levels is termed ODA.
- **Operational Data Analytics Visualisation (ODAV)**: The analysis performed on monitoring metrics needs to be visualised and presented with meaningful analysis such that it can used by the system operators and other stakeholders.
- **Operational Data Analytics Control (ODAC)**: Once the analysis results are reflected, the gained insights are then transferred back to the infrastructure for tuning

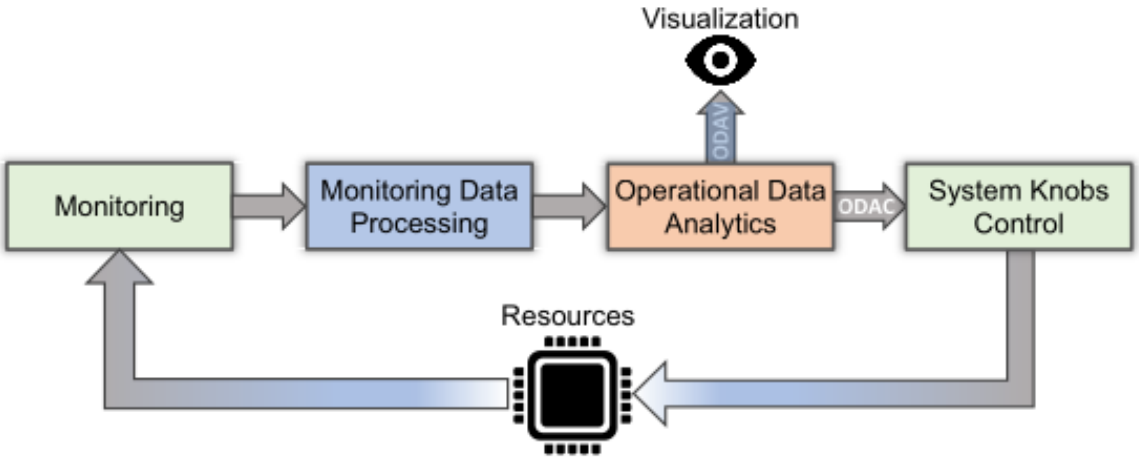


Figure 4.1: Basic stages composing a generic ODA pipeline (Source: [57]).

the system components for optimal energy consumption and optimal performance to bolster sustainability goals.

We start with the goal of creating an ODAV framework for OpenDC as part of this thesis project to be precise, which could be upgraded to have integrated ODAC capabilities in future.

4.2 Requirements analysis

Before proceeding with the design of an ODA framework, we need to go through the set of requirements which an ODA framework should meet. These requirements are essential for developing an effective ODA framework that can support the efficient operation of a data centre. These are categorised into functional and non-functional requirements.

Functional Requirements

These requirements specify the functions or features a system (design) must meet. As part of designing, we discuss the functional requirements below:

- **FR1 - Data persistence**: The ODA framework system design should include storage of exported metrics to a time-series database which could persist the data for both in-band and out-of-band analysis.
- **FR2 - Monitoring**: The ODA framework should continuously collect and enable analysis of data related to various data centre components within OpenDC (CPU utilisation, power usage, number of guests, etc.).
- **FR3 - Analysis**: It should enable an analysis of the energy and performance-related metrics available in the time-series database.
- **FR4 - Configurability**: Easier configuration of the host environment is required to allow experimentation with varied host environments.
- **FR5 - Flexibility**: It should be flexible enough to analyse the energy consumption of multiple scheduling policies to highlight better energy-efficient task scheduling algorithms available in the portfolio with respect to various use cases.

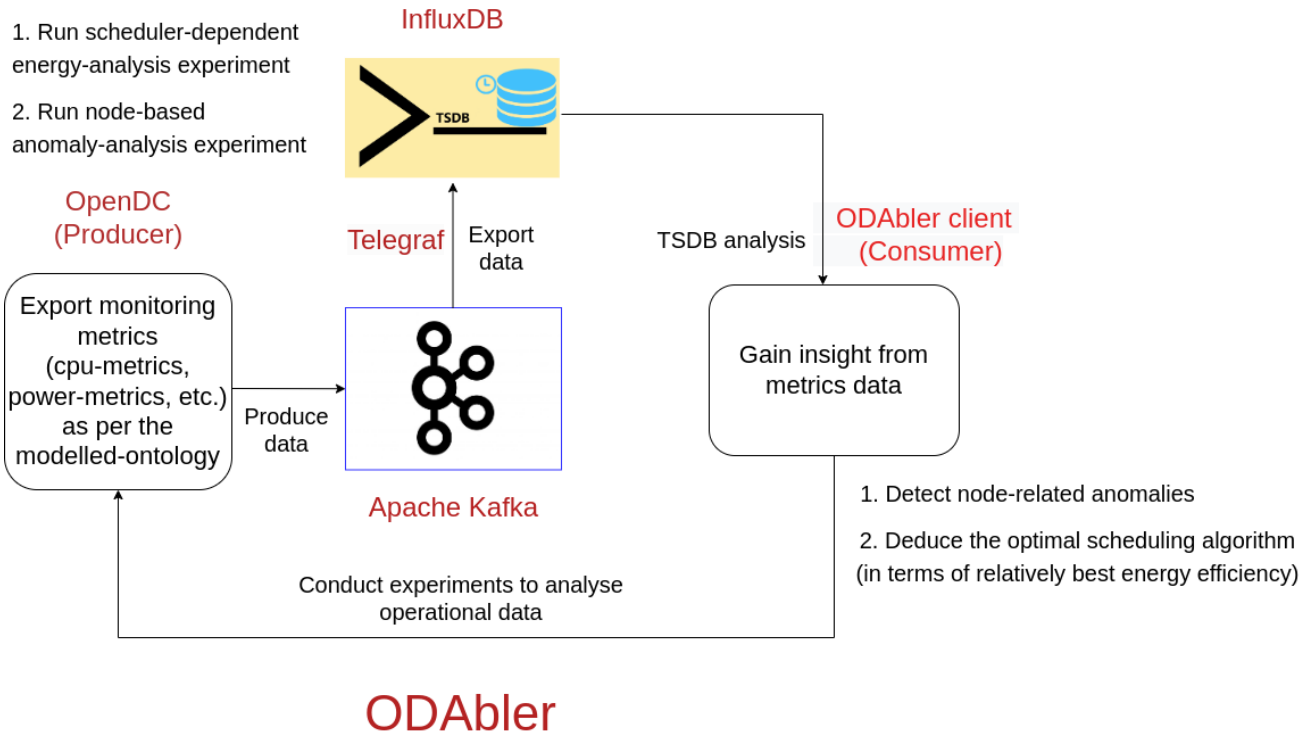


Figure 4.2: High-level architecture of ODAbler framework showing various components and its integration with OpenDC

Non-Functional Requirements

Non-functional requirements are those requirements which define the quality attributes or constraints a system must adhere to. These requirements describe how the system should adhere to in terms of certain functionalities. Below we discuss some of the non-functional requirements which have been considered as part of our design goals.

- **NFR1 - Scalability**: The ODA framework must be scalable to accommodate the growing amount of data and devices in the data centre.
- **NFR2 - Reliability and Availability**: The ODA framework should be reliable and available, ensuring that it can operate continuously withstanding failures at the data centre side.
- **NFR3 - Performance**: It should be capable of processing and analyzing data in a timely manner, providing near real-time insights wherever possible to support proactive management of the data centre's operations.

4.3 Proposed Architecture

The key design element involves several key elements as represented in Fig. 4.2. Some of the key elements include OpenDC itself, and Apache Kafka as the middleware which is responsible for message publishing to a time-series database 'InfluxDB', using the 'Telegraf' agent. Once the time-series data is available at InfluxDB, the ODAbler client application performs out-of-band analysis (as part of the current scope of its implementation). The key technical components (other than OpenDC that has already been described earlier)

are dedicated described below:

- **Apache Kafka**: Apache Kafka is an open-source distributed event-streaming platform used by several organisations for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications[1] [58, 59].
- **InfluxDB**: It is a time-series database built specifically for time-series data[2] [60]. Its key features include native SQL support, low-latency queries, superior data compression, single datastore support for all time-series data, openness and interoperability with data ecosystems, and unlimited cardinality.
- **Telegraf**: It is an open-source plugin-driven server agent that is responsible for collecting and sending metrics and events from various databases, systems, and IoT sensors[3] [61, 62]. It is written in Go and has no external dependencies due to its compilation into a single binary, which allows its easier execution on any system. It requires a very minimal memory footprint. There are 4 types of Telegraf plugins: input, process, aggregate and output. We use the input and output plugins in the Telegraf configuration, where the input plugin collects metrics from the Kafka system and the output plugin writes to the InfluxDB datastore.
- **OpenDC simulator**: OpenDC is a data centre simulation framework which allows various experimentations related to task scheduling and others [11]. This project's goal is to create a digital twin for OpenDC, showcasing the ODA capabilities using ODAbler on top of the OpenDC simulator.
- **ODAbler client (analyser) application**: A pure Python-based analysis application, which is responsible for communicating with OpenDC to trigger the execution of energy-efficiency-analysis and node-anomaly-analysis related experiments[4]. Once the experiments are executed in OpenDC, the data made available to InfluxDB is analysed in this ODAbler client application, meaningful plots are visualised and relevant data points indicate the result of the analysis. This ODA-related application is a novel contribution as part of this thesis project.

As presented in Figure 4.5, various ODA functionalities like "anomaly detection" (part of fault detection from layer 2), resource "allocation and scheduling" (from layer 3), "energy modelling" and "monitoring & alerting" (from layer 6), and layer 7 forms the crux of ODAbler experimentation using OpenDC (allowing 'reusability' and 'experimentation' of data centre operations).

→ The proposed architecture (consisting of proven technical components like Kafka, InfluxDB and Telegraf) satisfies the key requirements, some of which are discussed in the next couple of sections, and the experiments discuss the results to strengthen the claim.

4.4 Design elements overview

The key technical components discussed in the previous section are set up in a way that meets all of the requirements outlined in the requirements analysis section.

[1] Apache Kafka - `https://kafka.apache.org/`
[2] InfluxDB - `https://www.influxdata.com`
[3] Telegraf - `https://www.influxdata.com/time-series-platform/telegraf/`
[4] ODAbler analysis client - `https://github.com/am-i-helpful/ODAbler`

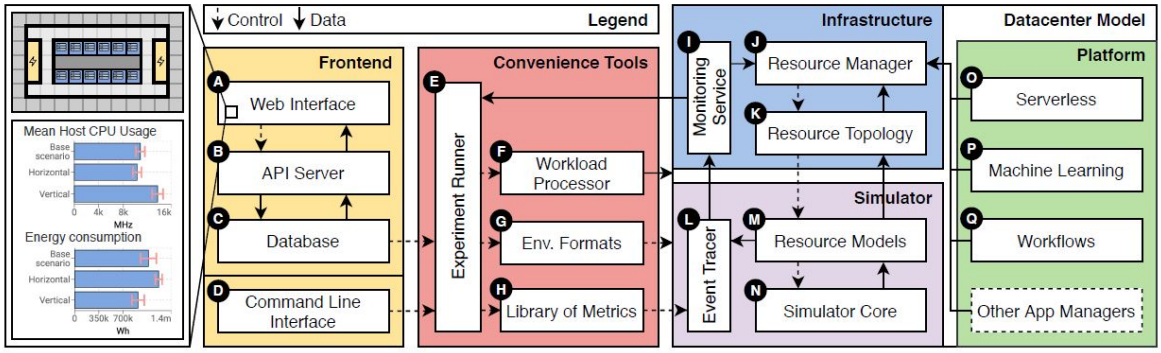


Figure 4.3: An overview of the architecture of OpenDC (2.0) (Source: [11]).

OpenDC 2.0 (or simply, OpenDC) operation

Please note that all references to OpenDC in this thesis project correspond to the enriched version of OpenDC, i.e., OpenDC 2.0. Fig. 4.3 shows an overview of the architecture of OpenDC. In this subsection, we discuss the functionalities of key classes which are available within OpenDC 2.0, and play a crucial role in executing energy-related experiments in OpenDC. From the operational perspective, the OpenDC simulator allows the simulation of workload trace scheduling on a cluster (datacentre) environment configured with the desired host specifications. Although OpenDC features are heavily documented within the codebase, the author felt that there is a lack of description of the functionalities which could simplify the complexity of understanding OpenDC functionalities for a new user. As a contribution to the same, we have shared our understanding of the OpenDC simulator from a technical guide perspective, in addition to the existing OpenDC documentation by the authors of OpenDC. Some of the key classes around which the ODA framework (ODAbler) design revolves are discussed below. Please note that the classes starting with 'Sim' represent that is a simulated representation of the corresponding real-world entity.

- **WorkflowService**: An interface that defines the service for workflow execution. This interface specifies an abstract invoke() function using the WorkflowService implementation elements (Dispatcher, ComputeClient, Job AdmissionPolicy, JobOrderPolicy, TaskEligibilityPolicy, and TaskOrderPolicy).
- **WorkflowServiceImpl**: This class implements the WorkflowService interface, by adding invoke() function implementation details besides other functions. The class provides an implementation for various stages specified in the "reference architecture for datacentre scheduling" literature [9]. Stages like J1 (incoming jobs), J2 (creation of a list of eligible jobs), J4 (performing certain actions per job), T1 (creating a list of eligible tasks), and T3 (performing certain actions per task) are implemented in various functions in the class. This class is critical and various additional configurations can be done on a central level at the scheduler using this (e.g., fault injection boolean flag is configured for controlling node-anomaly experiments related to the ODA framework design).
- **WorkflowServiceTest**: This is a test class which allows the execution of the workload trace, and allows verification that all tasks in a trace are executed correctly. This is the integration-test-related class that should be explored first in OpenDC for understanding the trace file execution and energy consumption-related operations.
- **ComputeMonitor**: This interface specifies a monitor that tracks the metrics and events of the OpenDC compute service. It has 3 record() functions, one each for

Server, Host and Service related compute recordings. A class should implement this interface for retrieving host or server records (metrics), as done in the ODAComputeMonitor class during ODAbler framework implementation.

- **HostTableReader**: This interface provides the declaration of various host-related metrics (host, timestamp, guestsRunning, cpuUtilisation, powerUsage, etc.) which can be extended.
- **ComputeService**: This is one of the most critical classes in the OpenDC simulator that hosts the implementation of the OpenDC compute engine.
- **MonitoringMetrics**: A special data class which defines the metrics to be exported to Kafka, driven by the modelled ontology of HPC cluster(s). Some of the key metrics elements used in the experiments are powerUsage, energyUsage, serverId, timestamp, policyId, and guestsRunning.
- **OpenDCNodeConfig**: A newly added configuration class in the newer "opendc-oda-experiments" module (implemented as part of this project) which contains the specification of cluster host environment (describing configurations of CPU, memory, etc. in the host).
- **ODAExperimentListener**: A newly added class in the newer "opendc-oda-listener" module (implemented as part of this project) enabled to start a server socket, which allows external ODAbler client analyser to connect requesting to run the experiment(s).
- **ODAComputeMonitor**: This ODA class extends ComputeMonitor and triggers the export of ontology-modelled metrics to Kafka at every defined interval when the record() function is triggered.
- **ODAExperimentKafkaProducer**: The ODA class which statically initialises Kafka producer instance (using the "kafka.clients" library) that assists in sending messages to the Kafka platform.
- **SchedulingAlgorithmComparatorExperiment**: This ODA class executes the energy-efficiency experiment collecting and exporting ontology-modelled metrics during the execution of various task-scheduling algorithms implemented in OpenDC.
- **NodeAnomalyIdentifierExperiment**: The ODA class that executes the node-anomaly experiment collecting and exporting ontology-modelled metrics during the execution of various task-scheduling algorithms implemented in OpenDC.
- **HostSpec**: This class describes a physical host's specification that is to be simulated within OpenDC, which will host the virtual machine(s) internally.
- **TraceHelpers**: This class consists of a helper method which replays the trace on the list of virtual machines (VMs), and suspends execution until all VMs have finished their execution.
- **Job**: A class representing a DAG (directed acyclic graph) of tasks with control and data dependencies between tasks.
- **Task**: A class that represents the lowest level element of a workload, which is basically a stage of a job that is scheduled by the task-scheduling algorithms.

- **TaskState**: This class declares various states (created, started, finished) of the tasks which are scheduled for execution.
- **Host**: This interface declares various methods about the host specifications and functionalities (including the fault injection capability).
- **SimHost**: This class implements the Host interface describing the host specifications and functionalities. This class allows the faults to be injected, if the experiment is configured with the boolean flag related to node anomaly set as true.
- **SimHypervisor**: This class is another critical class which facilitates the execution of multiple concurrent workloads, while acting as a single workload to another machine.
- **SimHypervisorCounters**: This class represents the performance counters of a SimHypervisor.
- **ProcessingUnit**: This class represent a single logical compute unit of processor, that could be either virtual or physical.
- **SimBareMetalMachine**: This class acts as a simulated bare-metal machine that executes a single workload.
- **SimPsu**: This interface represents a power supply configuration, which enables the construction of a power supply for the SimMachine interface.
- **SimMachine**: This interface represents a generic machine that is able to execute workload objects.
- **SimPsuFactory**: An interface that defines a SimPsu unit for the specified SimMachine.
- **ComputeMetricReader**: This is an important helper class that collects metrics for a ComputeService instance, and exports it automatically at every export interval.
- **Provisioner**: It is a helper class that sets up the experimental environment in a reproducible manner, in which the users configure the environment using multiple provisioning steps which are executed sequentially.
- **ProvisioningStep**: This interface declares a provisioning step having functions apply() and create() which are responsible for provisioning various infrastructure elements as required in the simulation experiment.
- **ComputeMonitorProvisioningStep**: This critical class provisions a ComputeMetricReader to periodically collect the metrics (as defined by export-interval) of a compute service, and report them to a ComputeMonitor implementation.
- **ComputeServiceProvisioningStep**: This class provisions a compute service without hosts, including the construction of a compute scheduler.
- **ComputeSteps**: This is another critical class which contains functions to setup compute service, register compute monitor, and setup hosts, all of which form the backbone of executing the workload trace on a host environment.
- **HostsProvisioningStep**: This class is responsible for provisioning a list of hosts to be used by the compute service.

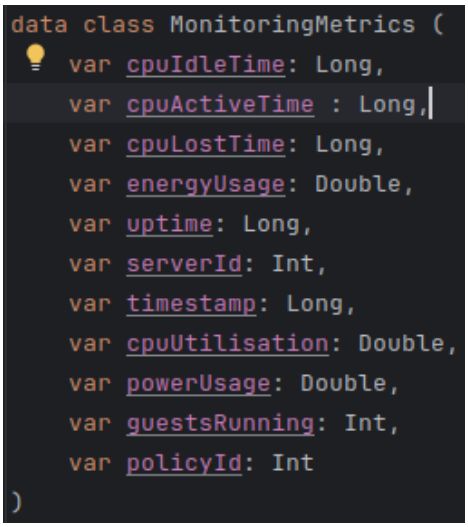


Figure 4.4: Metrics exported to time-series database reusing the concepts from the modelled ontology (in the previous chapter).

- **WorkflowServiceProvisioningStep**: This class provisions a workflow service which manages the workflow execution.
- **Dispatcher**: An interface that defines a schedule to execute future tasks over multiple threads potentially.

Ontology-driven export of OpenDC metrics

As shown in Fig. 4.4, the corresponding attributes or properties are selected to be exported to the time-series database InfluXDB for further analysis by the ODAbler analysis client. This screenshot is taken from the OpenDC's MonitoringMetrics class, which is a data class whose sole purpose is to hold data of various metrics (as reflected by the name of the member properties). The member properties are briefly described below clarifying the context and their usage as part of the ODAbler project:

- **CPUIdleTime**: time spent by CPU in an idle mode on the host; this is not used in our analysis (but could be considered as part of future use and extension of the project)
- **CPUActiveTime**: time spent by CPU actively spending CPU cycles on the host; this is not used in our analysis (but could be considered as part of future use and extension of the project)
- **CPULostTime**: time lost by CPU due to interference on the host; this is not used in our analysis (but could be considered as part of future use and extension of the project)
- **EnergyUsage**: energy-usage of a host after a given timestamp; used in the energy-efficiency experiment to compare the task-scheduling algorithms
- **UpTime**: time since the host is up and running; this is not used in our analysis (but could be considered as part of future use and extension of the project)
- **ServerId**: the unique identifier of the host server (node) helping it get differentiated among other host servers (nodes) in the cluster; used in the node-anomaly experiment to uniquely identify nodes which were fault-enabled

- **Timestamp**: the timestamp of the workload (jobs/tasks) execution exported to Kafka (and InfluxDB); this is a critical property and is used in both types of experiments (part of the modelled ontology).

  → It is critical to note here that the InfluxDB's time-series timestamp is not being used for analysis in this project, but we rely on the timestamp being shared by OpenDC instead - the reason for the same is because the workload traces are being simulated and not actually running in a production environment. Thus, the simulation completes within 5 minutes (depending on different trace files and different scale configurations), making the timestamp-based index of the time-series database not useful for this project. Still, we want to portray a real-world scenario because in production environments, the metrics are exported in a time-series format to a time-series database (like) InfluxDB, and we still wanted to show how could we adhere to the production-like applications and practices, and at the same time, match the expectations with the simulated analysis. This is very critical to note for all such simulator-based experiments that the timestamp-indexing of data insertion to the database would not help, because the actions are being simulated very quickly (ranging from seconds to minutes depending on the workload trace file) when compared to the real-world production environments (where metrics are exported actually after a certain fixed interval, e.g., every 15/30 seconds).

- **CPUUtilisation**: reflects the CPU utilisation of a given host; this attribute is not directly used in ODA-related analysis, but this property is critical as it drives the power usage (and thus, the energy usage too)

- **PowerUsage**: the instant power usage value at a particular timestamp; used in the node-anomaly experiment to uniquely identify nodes which have fault-enabled power values

- **GuestsRunning**: total number of guests (VMs) running on the system; this attribute is used in the node-anomaly identification experiment, to find a mismatch between power values of the non-anomalous vs anomalous dataset, because for a given number of guests running on the host server, the power value should be identical.

- **PolicyId**: reflects the policy identifier used to uniquely identify the task-scheduling policy for calculating their energy efficiency respectively.

ODAbler client (analyser) overview

The ODAbler client (analyser) is a novel contribution as part of this thesis project. This application retrieves the OpenDC host environment metrics exported to InfluxDB via Kafka, and performs in-band and/or out-of-band mode analysis both online (near real-time) or on-demand (later), as per the requirement. Since the Telegraf agent might act as a bottleneck while writing several metrics to InfluxDB within a few seconds of simulation for hours of task trace execution, it is more practical to allow on-demand operation for analysis.

In the current application, the ODAbler client performs descriptive analysis on energy usage and power usage metrics, highlighting the energy efficiency or node anomaly as desired in respective experiments. The ODAbler client has a main() function which is configured to initiate the connection to OpenDC, requesting for an 'ENERGY' or 'ANOMALY' related experiment, or trigger a 'TERMINATE' signal for shutting down OpenDC's server socket (OpenDC's parent process). The ODAbler client also has a module to detect weather

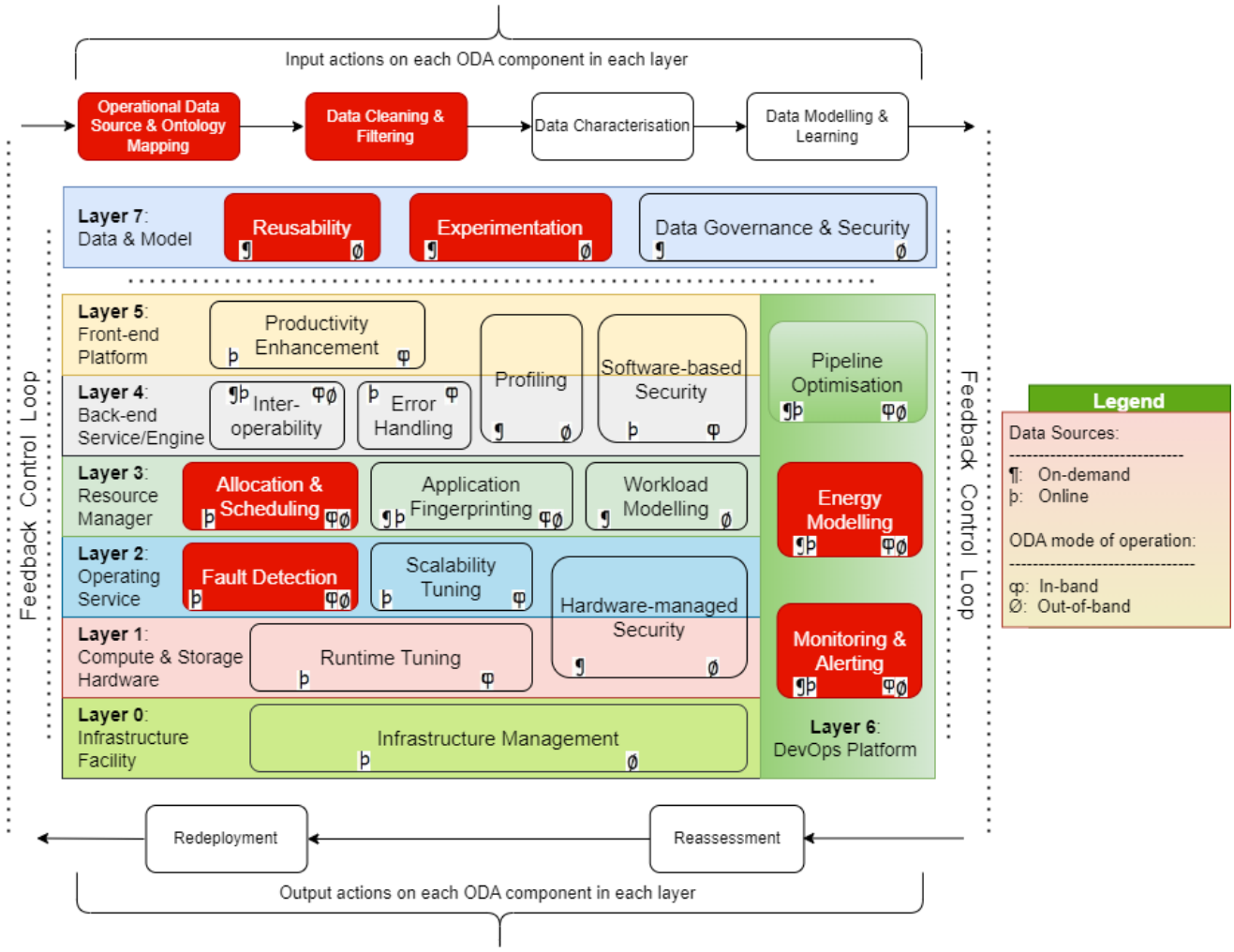


Figure 4.5: The implemented ODA capabilities in the ODA reference architecture are highlighted in red colour

metrics for the data centre location. Once OpenDC enables the export of data to the time-series database InfluxDB, the analyser templates are made available in the form of Python notebooks for various experimental configurations, and these notebooks are executed to analyse the metrics and publish the results in the form of "energy-efficiency" calculations and visualisations, and "node-anomaly" detection and visualisation (based on a suitable threshold value).

4.5 Summarised technical implementation

As mentioned earlier, ODA is needed at all levels of a distributed ecosystem, which relies directly on the monitoring of various data sources at different time scales. We consider two types of data sources, as identified by the Wintermute team in their ODA framework [31]:

- **In-band**: data that is sampled and consumed within a specific component at any layer of the distributed ecosystem. The underlying techniques operate at a fine temporal scale and require low analysis overhead and latency while collecting data.

- **Out-of-band**: data coming from any of the available sources in the system, including historical or asynchronous data. The underlying operation has to be performed at a coarse scale (in the order of minutes or higher) and must be synchronised explicitly, but latency and overhead are less of a concern in this scenario.

Additionally, we also categorise ODA techniques into two modes of operation as listed below, originally grouped by the Wintermute team [31]:

- **Online**: a continuous operation resulting in output resembling a time series, which can be (re)used at various levels as a feedback loop.
- **On-demand**: operation triggered at specifically scheduled times to steer decisions managing the information about the system's status.

→ The technical implementation of the ODAbler framework involves the following steps, encompassing various components outlined in section 4.3, and covering the ODA elements as shown in Fig. 4.5:

- **Enable fault injection in OpenDC**: The first step in the design of ODAbler is to enable fault injection in OpenDC. This is done via adding a boolean flag which controls whether the experiment is fault-driven or not. And, the fault is controlled via the SimPsuFactories class where we set the power usage value to the current value + 10 for multiple hosts.
- **Launch InfluxDB and Kafka**: InfluxDb docker image has been used in this project for the experiment, but a standalone database version would also work. Apache Kafka native local installation on Fedora (Linux) is used in the current implementation, and both Kafka and InfluxDB should be launched before launching any other application.
- **Start Telegraf service**: The Telegraf service has a configuration which has been tweaked to parse topics from Kafka (which is specific and modified as per project requirements) and write to InfluxDB in specific measurement and in a specific format (which is accordingly analysed by ODAbler client analyser application). This configuration (telegraf.conf) is available in the ODAbler repo, which should be placed at the desired location (/etc/telegraf/telegraf.conf) before starting the Telegraf service. Only when the Telegraf service is running successfully, further steps should be followed.
- **Launch OpenDC server**: The OpenDC simulator is enriched with a configuration that allows it to start a server socket (TCP) to accept socket connection requests from the ODAbler client application. Once the OpenDC server socket is up and running, only then the ODAbler client analysis application can be invoked, otherwise the client application would throw a socket connection refused error.
- **Launch ODAbler client analyser application**: The ODAbler client analyser application is enabled with the functionality to run either the energy-efficiency analysis experiment or the node-anomaly analysis experiment, or both sequentially, and initiate the termination of the OpenDC server connection at the end by sending a termination message.
- **Export operational data from OpenDC to Apache Kafka**: Execute the 'ENERGY' or 'ANOMALY' experiment (or, both sequentially) in ODAbler client application (from the main.py's main() function), which would trigger either the SchedulingAlgorithmComparatorExperiment or the NodeAnomalyIdentifierExperiment class in OpenDC for producing the metrics to Kafka.

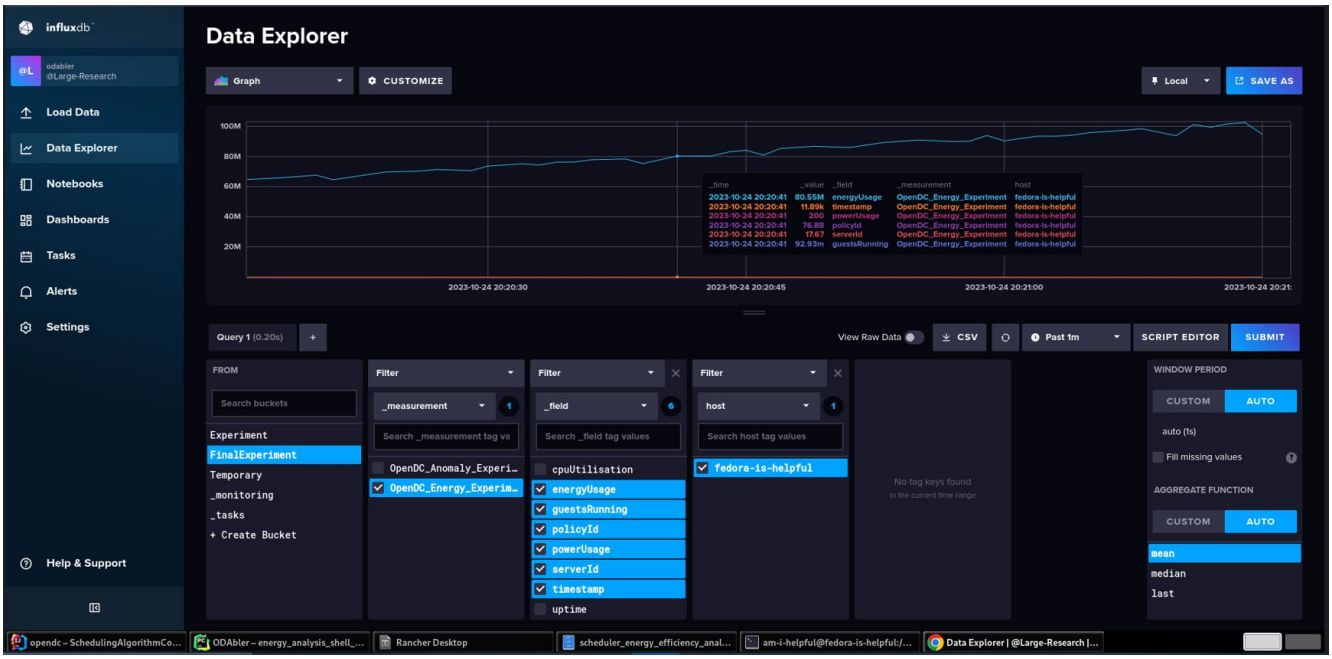


Figure 4.6: OpenDC exported metrics visualised by default in InfluxDB data explorer (however, it has limited capabilities concerning our analysis requirements).

- Kafka exports the ontology-driven relevant power usage and energy usage metrics (besides others) to InfluxDB via the Telegraf agent
- Data is persisted at InfluxDB with the help of the Telegraf agent, and this should be verified with the help of the online visualiser capability available (just to be sure that the data is written). Sometimes, the Telegraf agent might act as a bottleneck in writing various metrics locally to the InfluxDB taking longer than expected, due to the huge trace timestamp-based execution (as is the case with Pegasus_P2 trace in our experiment), which forces the use of the out-of-band mode of analysis for such trace files.
- ODAbler performs out-of-band analysis on InfluxDB data, once the data is fully available. In the current implementation, the node anomaly experiment is triggered automatically because it uses the shell industrial trace, which has a total execution time of tasks equal to 630 seconds, whereas the other (scientific) trace Pegasus_P2 has a total execution time of beyond 35 hours, which takes longer to be written to InfluxDB and slow query, and thus not used in the node-anomaly-analysis experiment.

A sample visualisation as reflected in the data explorer menu in the InfluxDB console is shown in Fig. 4.6.

Task-scheduling algorithms' energy-efficiency comparison

OpenDC is a discrete-time data centre operations simulator, thus, there is no functionality to actually measure the energy or power consumption at times when there is no workload for execution. Thus, the energy experiments are directly dependent on the workload trace file, and accordingly on the timestamp and other requirements as governed by the trace file.
As discussed in the previous section on OpenDC in this chapter, the OpenDC is enriched with data exporting capabilities to InfluxDB via Kafka (and Telegraf agent). The energy-efficiency analysis to be done later in the ODAbler client analyser application relies on

the 'energyUsage' parameter being exported as part of OpenDC metrics (driven by the modelled ontology), which signifies the total energy consumed up to a specific timestamp of execution of the workload trace. This is made available by the *HostTableReader* interface and *ODAComputeMonitor* class (which extends from *ComputeMonitor* class) that provides records related to various properties of all of the hosts configured as part of the cluster setup for the experiment.
The 'energyUsage' attribute is influenced by 'powerUsage' and the timestamp of trace execution. Whereas, the 'powerUsage' attribute is influenced by the total number of guests running on the host, and the CPU frequency (CPU utilisation) at which the guests are running. Once these records are exported to the InfluxDB, ODAbler could perform in-band or out-of-band analysis comparing the energy consumption of different task-scheduling algorithms, and determining the most efficient among them for different workload trace files and different cluster configurations. The aim of the energy-efficiency detection experiment is to highlight the task scheduling algorithm which consumes the least amount of energy for executing tasks in trace for the given data centre cluster configuration. The energy-efficiency analysis part is discussed in detail in the next chapter in its respective experiment section.

Anomaly injection in hosts' power usage

To achieve anomaly injection, the hosts' power usage needs to be modified subtly. The injection is made static, and during each time reading of record by the compute monitor class, an addition of 10 watts (W) of power is statically increased on two hosts with hostname "host-3" and "host-9".
These two hosts were chosen randomly for anomaly injection to inject anomaly on multiple hosts, and there is no particular reason as to why these hostnames were selected. These hostnames were statically injected with the anomaly in power usage, and a boolean flag 'isFaultInjected' (or, isAnomalousExperiment) configures whether the trace execution would run the anomalous power usage (thus, anomalous energy usage too). If the boolean flag is set to true, the trace execution would be injected with the anomaly in power usage value on both hosts, otherwise, if set to false, the execution would be uniform as earlier, and there would not be any anomaly in the power usage.
As part of the anomaly detection experiment, the aim of the experiment is to validate whether the designed ODAbler client analyser application is able to identify the node anomaly at various discrete intervals of time. It is extremely critical to understand that the power usage model selected as part of our experiment is selected as a cubic CPU power model because of better accuracy than other power models existing in OpenDC, as claimed in a work of literature analysing performance and energy consumption metrics [63]. The node-anomaly analysis part is discussed in detail in the next chapter in its respective experiment section.

4.6 Summary

In this section, we discussed the design of 'ODAbler', an ODA framework for a data centre simulator 'OpenDC'. We discussed the requirements as to why various components like Kafka, InfluxDB (and Telegraf) have been selected as members of the ODAbler ODA framework. In the current approach, the design of ODAbler is geared towards seamless integration with OpenDC, but we could easily configure ODAbler in a way that it could also be integrated into other data centre simulators with some tweaking (given that it gives instructions to OpenDC to execute the experiments for producing the data using Kafka

producer client, and query data directly from InfluxDB for analysis purposes).

ODAbler is enriched with the in-band ODA mode of analysis of energy consumed by different task scheduling algorithms, and can easily detect node anomaly in an out-of-band mode of operation based on their power usage profile (as part of current implementation and scope). Further, both energy-efficiency calculation and node-anomaly detection can be experimented with using online or on-demand ODA techniques, once the data is available in InfluxDB.

Chapter 5

Results: Experimental evaluation of 'ODAbler'

In this chapter, we present the results and detailed discussion of various experiments conducted (in two categories) as part of the ODAbler framework design: (i) the energy-efficiency-based evaluation for multiple task-scheduling algorithms used for scheduling the workload tasks, and (ii) the node-based anomaly detection in terms of anomalous power-consumption value at specific time instants (originally injected via OpenDC on multiple nodes). A more summarised discussion of these results can be found in the succeeding sections.

5.1 Experimental setup

We set the experiment goals (EG) for this chapter by providing a general guideline which drives our experiment design and its execution, as listed below:

- **EG1** - Verification that the OpenDC simulator produces modelled ontology-driven energy and performance metrics (required for the energy and anomaly experiments).
- **EG2** - Comparing the power consumption for three different task-scheduling algorithms in a data centre environment
- **EG3** - Anomaly detection is successful and sufficiently visualised for the stakeholders to examine the same
- **EG4** - The experiments should be reproducible and the results should be consistent in multiple executions of the same experiment configuration.

With the goals set for the experiments, next, we discuss the experimental setup for conducting the experiments. As outlined previously in chapter 4, the ODAbler design involves a multitude of components performing specific roles required for the insight-based analysis. We first discuss the experimental setup before discussing the results of the experiment.

Experiment overview

As part of ODAbler design, it is enriched to analyse the energy efficiency of various task-scheduling algorithms which are implemented in OpenDC. It is also enabled with the capability to analyse anomalies for abnormal power usage on multiple nodes at discrete time instants. These two experiment categories serve as the overview, where the experimental evaluations are further elaborated in their respective sections. Section 5.2 corresponds to

Trace-file detail			
TraceID	Domain	Workflows	Tasks
shell[1]	Industrial	3,403	10,208
Pegasus_P2[2]	Scientific	5	5,813

Table 5.1: List of workload trace files (available on WTA archive) that were used for the experiments in ODAbler

the experimental evaluation of ODAbler in terms of analysing the energy efficiency of various task scheduling algorithms whose implementation is done within OpenDC. Section 5.3 corresponds to ODAbler's evaluation as an anomaly detection and visualisation framework highlighting the abnormal power usage on multiple hosts which are originally injected with faults within the OpenDC simulator.

Workload

Two workloads are used as part of multiple experiments as shown in Table 5.1: shell (industrial) [64] and Pegasus_P2 (scientific) [65]. These special workloads are actually categorised as workflows, as their set of tasks have precedence/data constraints between their respective tasks. An easier way to understand this is to think that the output of a certain task could be the input of another, thus creating a dependency between these two tasks.
The AtLarge Research group has collected several publicly available real-world workflows which have been packed into traces, all of which have been made available with their Workflow Trace Archive initiative[3] [66]. Both of the workloads have actually been referred from this WTA service, under the supervision of the head supervisor.

Compute environment (configured in OpenDC)

The computing environment is identical to that of the DAS-6 cluster[4] available at VU. The scaling of the hosts is set to 60% (representing 0.6 times the number of hosts in the VU cluster), 100% (meaning an identical configuration as the VU cluster), and 150% (referring to 1.5 times the number of hosts compared to the cluster used by VU). The resulting host count is rounded to the nearest integer, following the scientific convention/

Assumptions

OpenDC simulator is a discrete-time simulator, and thus, it does not accurately reflect power or energy consumption at a continuous range (which is also not required in the project). Also, the power or energy modelling is not affected by memory usage, storage, or other network behaviour (this behaviour is not configured in the current version of the OpenDC simulator).

Limitations

The experiments are executed on a laptop system which is quite old (age $>$ 5 years) and has a lower CPU configuration, which makes the data writing to InfluxDB slow for traces with larger scheduling time like Pegasus_P2, and as a result, the analysis at the ODAbler requires to wait for analysis until the energy-related metrics are available at InfluxDB.
The data retrieval for Pegasus_P2 trace is also affected by the system's old-age problem, and takes a longer time to retrieve data from InfluxDB. However, this latency limitation only affects the time taken for ODAbler to begin the analysis. This limitation doesn't affect the experimental evaluation results of ODAbler anyway though.

[3]WTA - `https://wta.atlarge-research.com/`
[4]DAS-6 clusters - `https://www.cs.vu.nl/das/clusters.shtml`

5.2 Scheduling policies' energy-efficiency analysis

This study investigates the energy consumption of three task-scheduling algorithms (FIFO, HEFT, and Random) when a certain workload trace is executed for a specific host-environment configuration in a data centre simulator. The primary objective of this set of experiments is to identify the most energy-efficient algorithm for executing workloads in a specific cluster environment that consumes the least amount of energy (among the three task-scheduling algorithms). The total energy consumption is measured, and these task-scheduling algorithms are compared to determine the energy efficiency. Negative percentages are used to flag the less efficient algorithms (in terms of energy consumption). The results are obtained accordingly which provide insights into optimising energy usage in data centre operations.

Methodology

• Each task-scheduling algorithm is configured to execute the same workload and cluster environment configuration in the data centre simulator OpenDC.
• The total energy consumption for each algorithm is recorded for comparison to find the most efficient one.
• Negative percentages are calculated to indicate the relative efficiency of these task-scheduling algorithms concerning energy consumption. A negative percentage denotes less efficiency compared to the most energy-efficient task-scheduling algorithm.
• The same process is repeated for another workload trace and a different scale of cluster configuration (with respect to the standard size of the DAS-6 cluster that is assumed as part of this project).

→ ODAbler uses both online and on-demand ODA mode of operation using in-band mode of analysis for energy-efficiency calculation. The operation could be online when the ODAbler client analyses the results soon after the data is exported from OpenDC to the time-series database InfluxDB, or it could be analysed later given that the timestamp of the start and end of the experiment is captured for use in the on-demand mode. Thus, it is important to note the time of execution of the experiment for performing on-demand analysis. As shell trace execution occurs quickly within 2 minutes (for all scales), we use online analysis there, whereas the Pegasus_P2 trace execution takes significantly longer to write data to InfluxDB (due to export interval set to 30 seconds for a 35-hour trace makespan), this analysis could be done in on-demand ODA.

A1. Using Shell industrial workload

The results of the experiment for executing the 'shell' industrial workload trace at different scales are summarised in Table 5.2. The task-scheduling algorithm with 0.00000 percentage efficiency shows the baseline for reference among the three task-scheduling algorithms for each block of cluster-scale configuration.

A1.1 - 60% cluster scaling: It is observed that when the cluster-scale configuration is set at 60% (near-rounded) of the standard DAS-6 cluster size, then the total energy consumed during the HEFT task scheduling algorithm's execution of the workload is the most optimal among the three. The total energy consumed in the case of the HEFT task-scheduling algorithm's trace execution is 0.734067 kWh, whereas the total energy consumption during FIFO and Random algorithms' trace execution are 0.734074 kWh and 0.734068 kWh respectively. Clearly, in this case, the total energy consumed is the least when the HEFT algorithm executes the workload trace, thus making it more efficient than

Task-scheduling algorithm total energy consumption - 'shell' trace			
Algorithm	Cluster-size configuration-percentage (%)	Total power consumption (in kWh)	Comparatively efficient (%)
FIFO	60	0.734074	-0.00099
HEFT	60	0.734067	0.00000
Random	60	0.734068	-0.00008
FIFO	100	1.201992	-0.00104
HEFT	100	1.201990	-0.00089
Random	100	1.201979	0.00000
FIFO	150	1.790465	0.00000
HEFT	150	1.790469	-0.00023
Random	150	1.790471	-0.00032

Table 5.2: Task-scheduling algorithms' total-energy consumption values for executing shell trace in OpenDC for different scale of host-environment (when compared to standard-size of DAS-6 cluster with 34 hosts; 100% means exact identical configuration, 60% means 0.6 times of cluster-size, and 150% means 1.5 times the size of the cluster respectively)

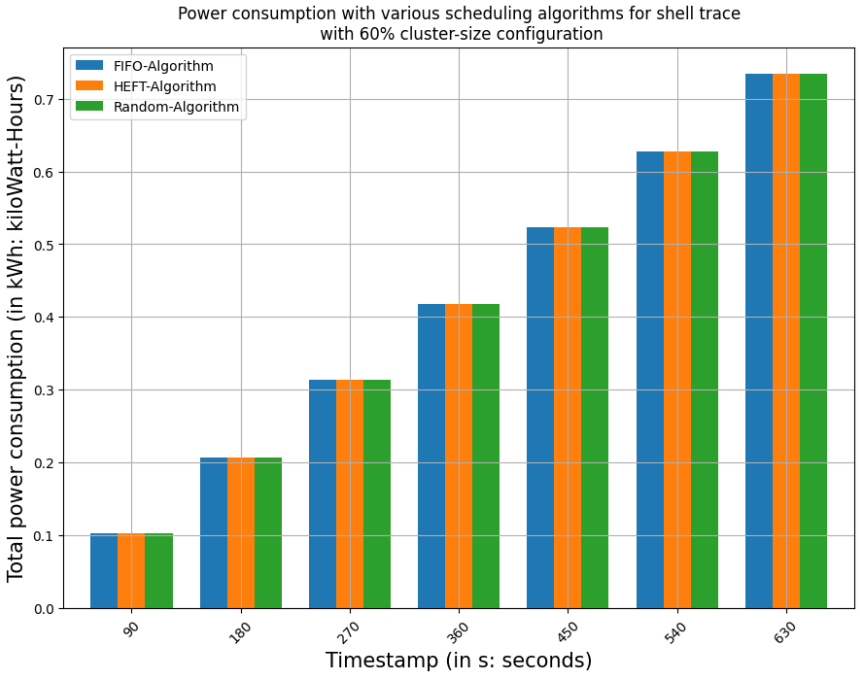


Figure 5.1: Task-scheduling algorithms' energy-consumption analysis at different time intervals of shell trace execution with cluster-size equal to 60% of the standard DAS-6 cluster size.

the other two (albeit very slightly). The energy-consumption analysis for this scenario is visualised in Figure 5.1.

A1.2 - 100% cluster scaling: When the cluster-scale configuration is identical to the standard DAS-6 cluster size (set to 100%), the total energy consumed during the HEFT task scheduling algorithm's execution of the workload is most optimal among the three

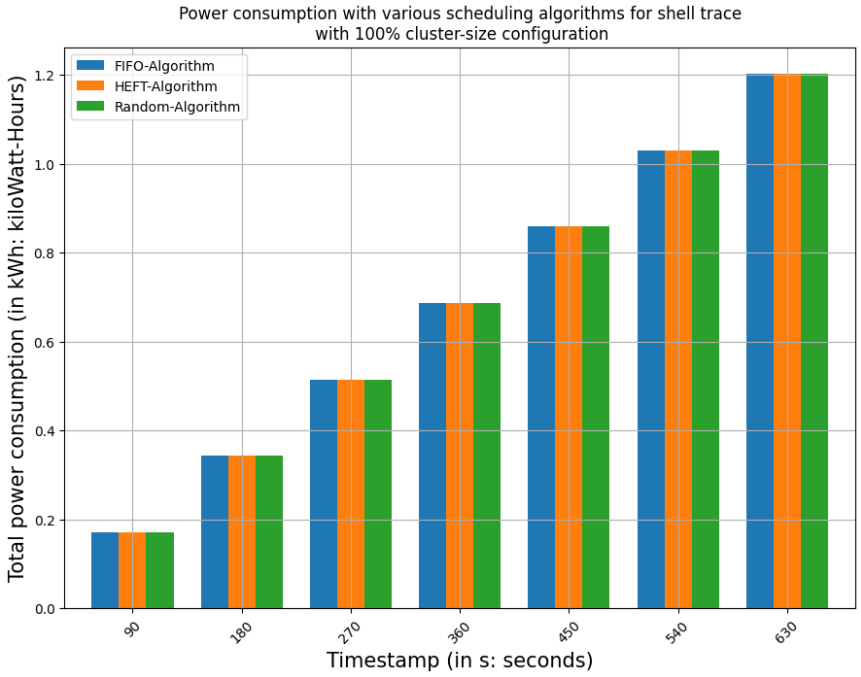


Figure 5.2: Task-scheduling algorithms' energy-consumption analysis at different time intervals of shell trace execution with cluster-size equal to 100% of the standard DAS-6 cluster size

(referring from the Table 5.2). The energy-consumption analysis for this scenario is visualised in Figure 5.2, which is difficult to contrast as the values are much closer to each other.

A1.3 - 150% cluster scaling: When the cluster-scale configuration is set to 1.5 times the standard DAS-6 cluster size (set to 150%), the total energy consumed during the FIFO task scheduling algorithm's execution of the workload is the most optimal among the three (referring from the Table 5.2). The energy-consumption analysis for this scenario is visualised in Figure 5.2, which is difficult to contrast as the values are much closer to each other.

A2. Using Pegasus2 scientific workload

The results of the experiment for executing Pegasus_P2 workload trace at different scales are summarised in Table 5.3.

The results of the experiment for executing the 'Pegasus_P2' scientific workload trace at different scales are summarised in Table 5.3. The task-scheduling algorithm with 0.00000 percentage efficiency shows the baseline for reference among the three task-scheduling algorithms for each block of cluster-scale configuration.

A2.1 - 60% cluster scaling: It is observed that when the cluster-scale configuration is set at 60% (near-rounded) of the standard DAS-6 cluster size, then the total energy consumed during the FIFO task scheduling algorithm's execution of the workload is most optimal among the three. The total energy consumed in the case of the FIFO task-scheduling algorithm's trace execution is 142.048 kWh, whereas the total energy consumption during HEFT and Random algorithms' trace execution are 142.161 kWh and 142.076 kWh respectively. Clearly, in this case, the total energy consumed is the least when the FIFO

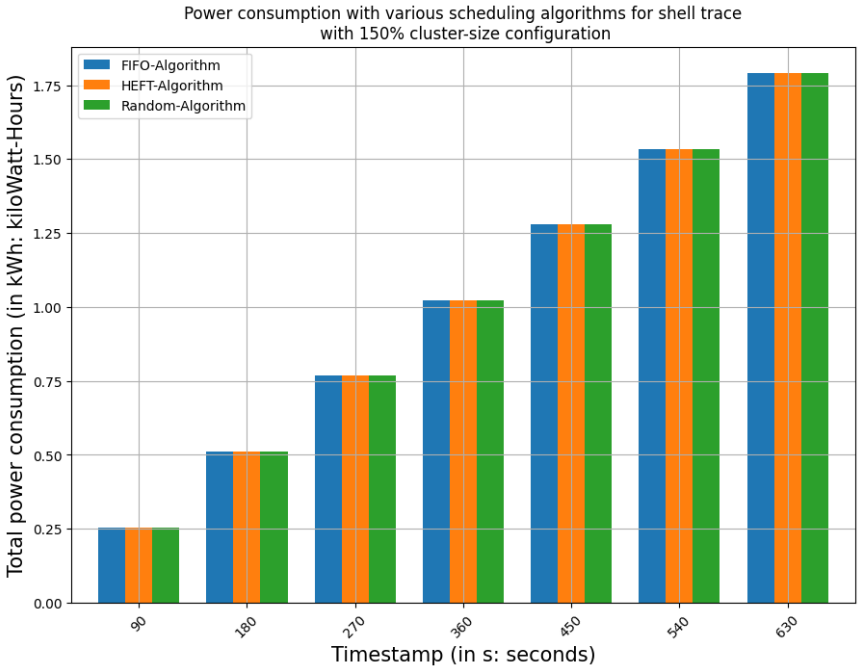


Figure 5.3: Task-scheduling algorithms' energy-consumption analysis at different time intervals of shell trace execution with cluster-size equal to 150% of the standard DAS-6 cluster size

algorithm executes the workload trace, thus making it more efficient than the other two (albeit very slightly). The energy-consumption analysis for this scenario is visualised in Figure 5.4.

A2.2 - 100% cluster scaling: When the cluster-scale configuration is identical to the standard DAS-6 cluster size (set to 100%), the total energy consumed during the HEFT task scheduling algorithm's execution of the workload is most optimal among the three (referring from the Table 5.3). The energy-consumption analysis for this scenario is visualised in Figure 5.5.

A2.3 - 150% cluster scaling: When the cluster-scale configuration is set to 1.5 times the standard DAS-6 cluster size (set to 150%), the total energy consumed during the Random algorithm's execution of the workload is most optimal among the three (referring from the Table 5.3). The energy-consumption analysis for this scenario is visualised in Figure 5.6.

Task-scheduling algorithm total energy consumption - 'Pegasus_P2' trace			
Algorithm	Cluster-size configuration-percentage (%)	Total power consumption (in kWh)	Comparatively efficient (%)
FIFO	60	142.048	0.00000
HEFT	60	142.161	-0.07973
Random	60	142.076	-0.01981
FIFO	100	240.060	-0.04015
HEFT	100	239.964	0.00000
Random	100	240.033	-0.02863
FIFO	150	359.066	-1.07187
HEFT	150	359.121	-1.08748
Random	150	355.258	0.00000

Table 5.3: Task-scheduling algorithms' total-energy consumption values for executing Pegasus_P2 trace in OpenDC for different scale of host-environment (when compared to standard-size of DAS-6 cluster with 34 hosts; 100% means exact identical configuration, 60% means 0.6 times of cluster-size, and 150% means 1.5 times the size of the cluster respectively)

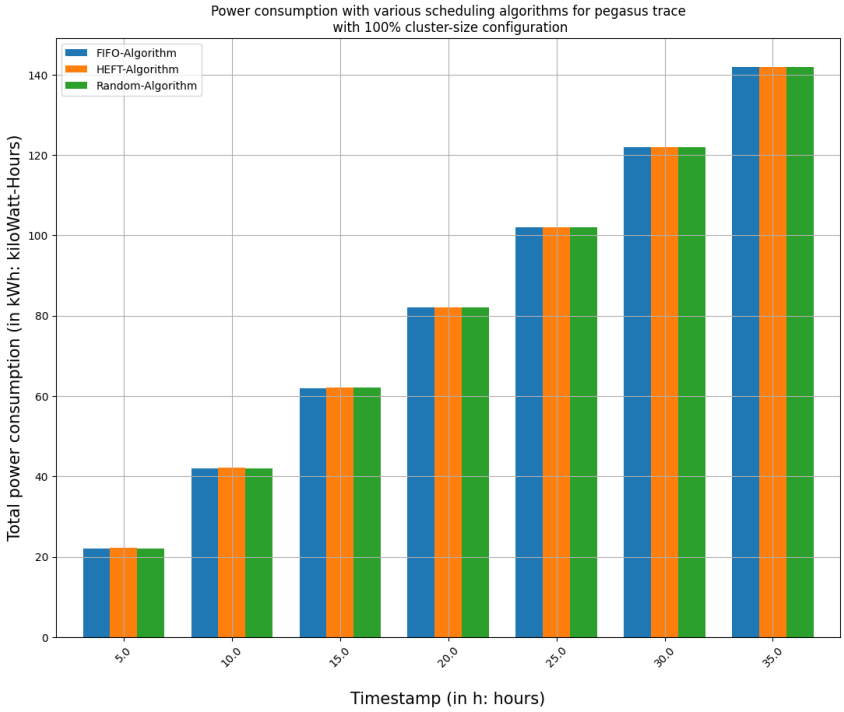


Figure 5.4: Task-scheduling algorithms' energy-consumption analysis at different time intervals of Pegasus_P2 trace execution with cluster-size equal to 60% of the standard DAS-6 cluster size

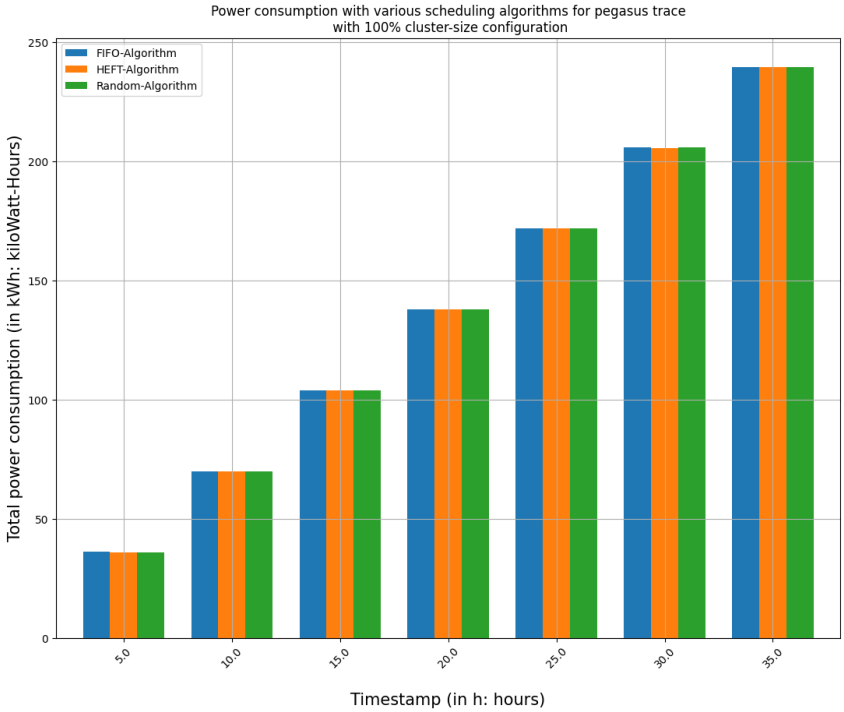


Figure 5.5: Task-scheduling algorithms' energy-consumption analysis at different time intervals of Pegasus_P2 trace execution with cluster-size equal to 100% of the standard DAS-6 cluster size

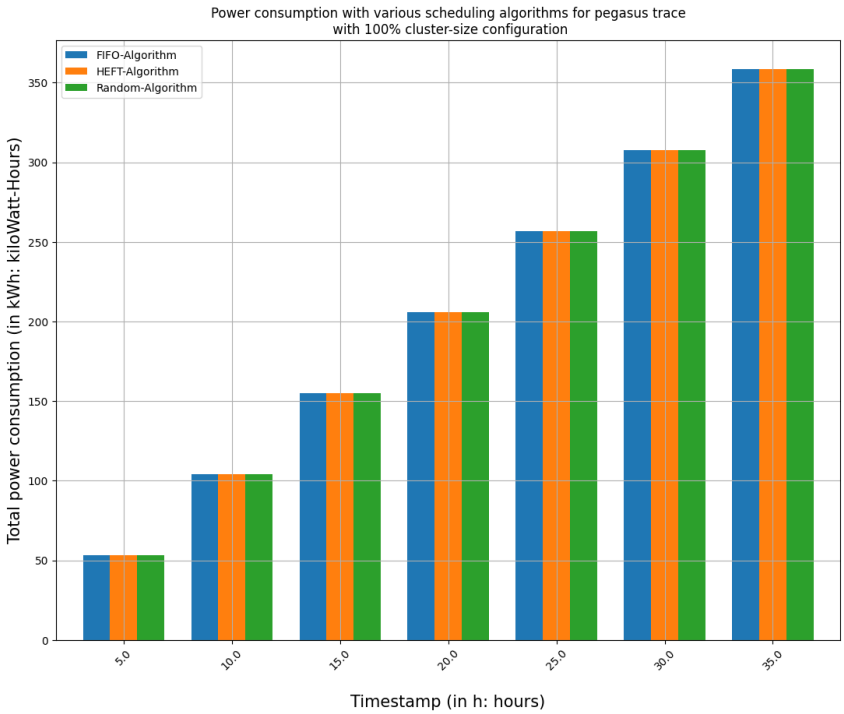


Figure 5.6: Task-scheduling algorithms' energy-consumption analysis at different time intervals of Pegasus_P2 trace execution with cluster-size equal to 150% of the standard DAS-6 cluster size

5.3 Node-based anomaly detection

This study presents a data-driven approach for detecting anomalies in power values within a network of nodes (which are available in a large-scale computing environment). The objective is to identify nodes (or servers, also called hosts) that exhibit unexpected variations in their power consumption compared to a reference dataset (which is known to have no anomalies). Please note that we have used the terms node, server or host interchangeably which represents a host configured for workload execution in a cluster setup, as all of them refer to the same concept. Although, in various research works of literature, apparently node-anomaly is comparatively a common term. The experiment has been conducted using a configurable boolean flag in OpenDC which allows a predetermined anomalous state to be enabled among a set of servers (which is discussed in the previous chapter in the design section), amongst the network of servers available in a large-scale computing infrastructure. The method involves merging two datasets: one representing the actual non-anomalous behaviour, and the other containing potential anomalies. Anomalies are detected in the servers by calculating the absolute difference in power values between corresponding records in the two datasets and applying a predefined threshold. Servers with power value differences exceeding the threshold are flagged as anomalous. The threshold can further be experimented with and reasoned for a better value, which is part of future work. In OpenDC, we inject faults on the hosts with host-id equal to "host-3" and "host-9" to produce anomalous power usage (having a difference of more than 10 watts), which will be identified as part of this set of experiments.

Methodology

Data Preparation: Two datasets are prepared - one representing the actual behaviour of server power values (in the absence of any anomalous setting) and the other containing potentially anomalous records.
Data Merging: The datasets are merged based on common attributes (e.g., time, server ID, and guests).
Anomaly Detection: The absolute difference between power values in the merged dataset is calculated. A threshold is applied to determine whether a server exhibits anomalous power consumption.
Results: Servers exceeding the threshold are identified as anomalous, and the results are reported.

The above method provides a straightforward approach to highlighting servers with anomalous power consumption behaviour. The choice of threshold is critical which should conform to the expectations of the production data centres, and thus, should be based on domain knowledge and the specific requirements of the analysis. While this approach may not be tied to a specific statistical test, it serves as a valuable exploratory tool for identifying potentially problematic servers within the configured environment. In terms of mathematical description, the approach can be mathematically represented as follows:
• Let P_non_anomaly be the power value in the reference (non-anomalous) dataset.
• Let P_anomaly be the power value in the dataset containing potential anomalies.
• Calculate the absolute difference |P_non_anomaly - P_anomaly| for each corresponding record.
• Define a threshold T to classify records as anomalous if |P_non_anomaly - P_anomaly| $>$ T.
This mathematical representation captures the essence of the approach. The choice of threshold is set to the numerical value of '10' because this value is injected as a fault for

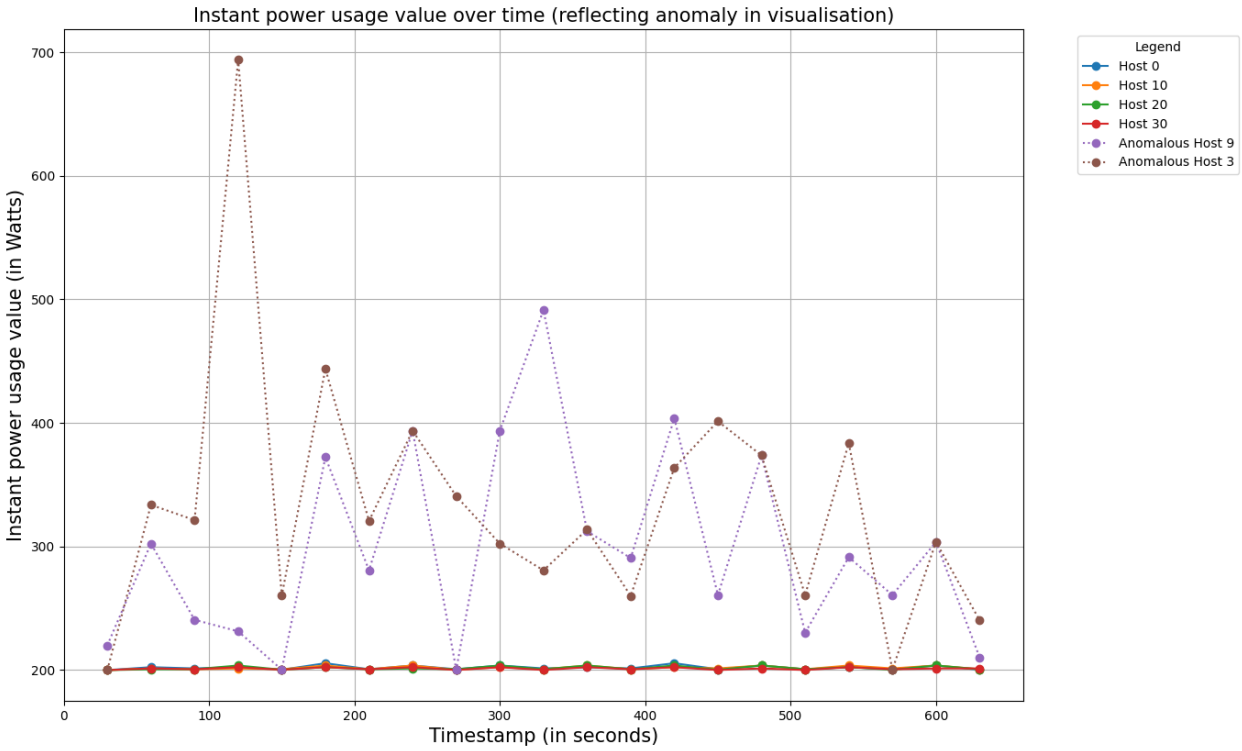


Figure 5.7: Node-based anomaly detection for abnormal power usage at different time intervals of trace execution with cluster-size equal to 100% of the standard DAS-6 cluster size

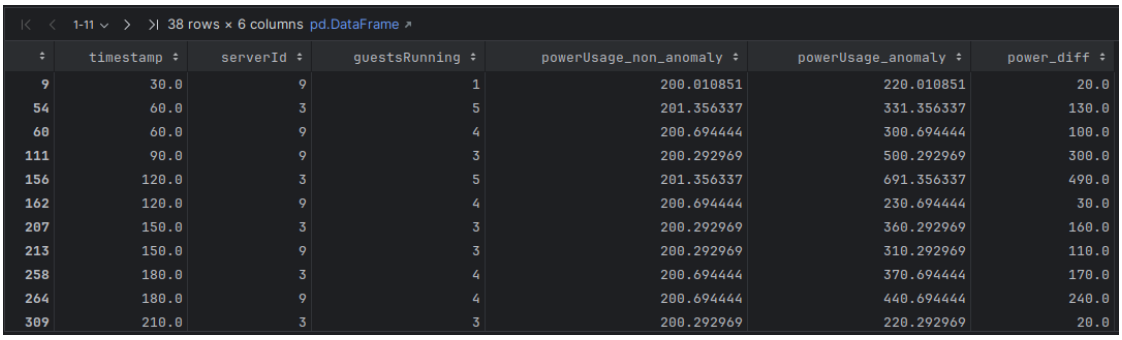
1-11 37 rows × 6 columns pd.DataFrame

	timestamp	serverId	guestsRunning	powerUsage_non_anomaly	powerUsage_anomaly	power_diff
9	30.0	9	1	200.010851	220.010851	20.0
37	60.0	3	7	203.721788	333.721788	130.0
43	60.0	9	6	202.343750	302.343750	100.0
71	90.0	3	5	201.356337	321.356337	120.0
77	90.0	9	4	200.694444	240.694444	40.0
105	120.0	3	7	203.721788	693.721788	490.0
111	120.0	9	5	201.356337	231.356337	30.0
139	150.0	3	3	200.292969	260.292969	60.0
173	180.0	3	7	203.721788	443.721788	240.0
179	180.0	9	6	202.343750	372.343750	170.0
207	210.0	3	4	200.694444	320.694444	120.0

Figure 5.8: Snippet taken from the ODAbler client's anomaly analysis experiment, where the details of timestamps when the node anomalies are flagged are also listed; this involves the experimentation with 100% cluster scale equivalent to the scale of DAS-6 cluster.

each compute monitor recording in the power usage at OpenDC (**Threshold, T = 10**). The interpretation of the results is discussed further in their respective experiment subsection.

→ ODAbler uses both online and on-demand ODA mode of operation using in-band mode of analysis for anomaly detection. The operation could be online when the ODAbler client analyses the results soon after the data is exported from OpenDC to the time-series database InfluxDB, or it could be analysed later given that the timestamp of the start and end of the experiment is captured for use in the on-demand mode. Thus, it is important to note the time of execution of the experiment for performing on-demand analysis.



1-11 38 rows × 6 columns pd.DataFrame

	timestamp	serverId	guestsRunning	powerUsage_non_anomaly	powerUsage_anomaly	power_diff
9	30.0	9	1	200.010851	220.010851	20.0
54	60.0	3	5	201.356337	331.356337	130.0
60	60.0	9	4	200.694444	300.694444	100.0
111	90.0	9	3	200.292969	500.292969	300.0
156	120.0	3	5	201.356337	691.356337	490.0
162	120.0	9	4	200.694444	230.694444	30.0
207	150.0	3	3	200.292969	360.292969	160.0
213	150.0	9	3	200.292969	310.292969	110.0
258	180.0	3	4	200.694444	370.694444	170.0
264	180.0	9	4	200.694444	440.694444	240.0
309	210.0	3	3	200.292969	220.292969	20.0

Figure 5.9: Snippet taken from the ODAbler client's anomaly analysis experiment, where the details of timestamps when the node anomalies are flagged are also listed; this involves the experimentation with 150% cluster scale equivalent to 1.5 times the scale of DAS-6 cluster.

Exp-B1: Executing 'shell' industrial workload on a cluster having standard DAS-6 cluster size

The first experiment as part of node anomaly detection compares the instant power usage value of all hosts with respect to a cluster identical to the DAS-6 cluster size. The results are visualised in ODAbler as shown in Fig. 5.7, and the data points flagged by ODAbler related to the anomalous time instants are represented in Figure 5.8. The anomalous points are the points where there is a difference of more than 10 watts in power usage at any instant, when compared to the non-anomalous hosts enabled experiment.

Exp-B2: Executing 'shell' industrial workload on a cluster with 1.5 times the DAS-6 cluster size

The second experiment as part of node anomaly detection compares the instant power usage value of all hosts with respect to a cluster identical to the DAS-6 cluster size. The data points flagged by ODAbler related to the anomalous time instants are represented in Figure 5.9, and the results are visualised in ODAbler as shown in Fig. 5.10. Again, the anomalous points are the points where there is a difference of more than 10 watts in power usage at any instant, when compared to the non-anomalous hosts enabled experiment.

5.4 ODA analysis' discussion

The two types of experiments conducted with different scales and multiple workload trace files reflect the significance of adopting ODA techniques in data centre operations (and other related operations fields). Although the experimental evaluation conducted as part of this project might appear to some as having a lack of in-depth analysis of key ODA techniques, it should be noted that this is the first step of designing an ODA framework for a data centre simulator. This is just the beginning of enabling more flexible experimentation for research in the field of ODA, similar to other research works carried out in the past using the OpenDC simulator.

Energy-efficiency analysis

This set of experiments carried out in Section 5.2 reflect a nice introduction to the capabilities of 'ODAbler' as an ODA framework. We notice that the results convey a unique summary that there is no clear winner in terms of the best task-scheduling algorithm

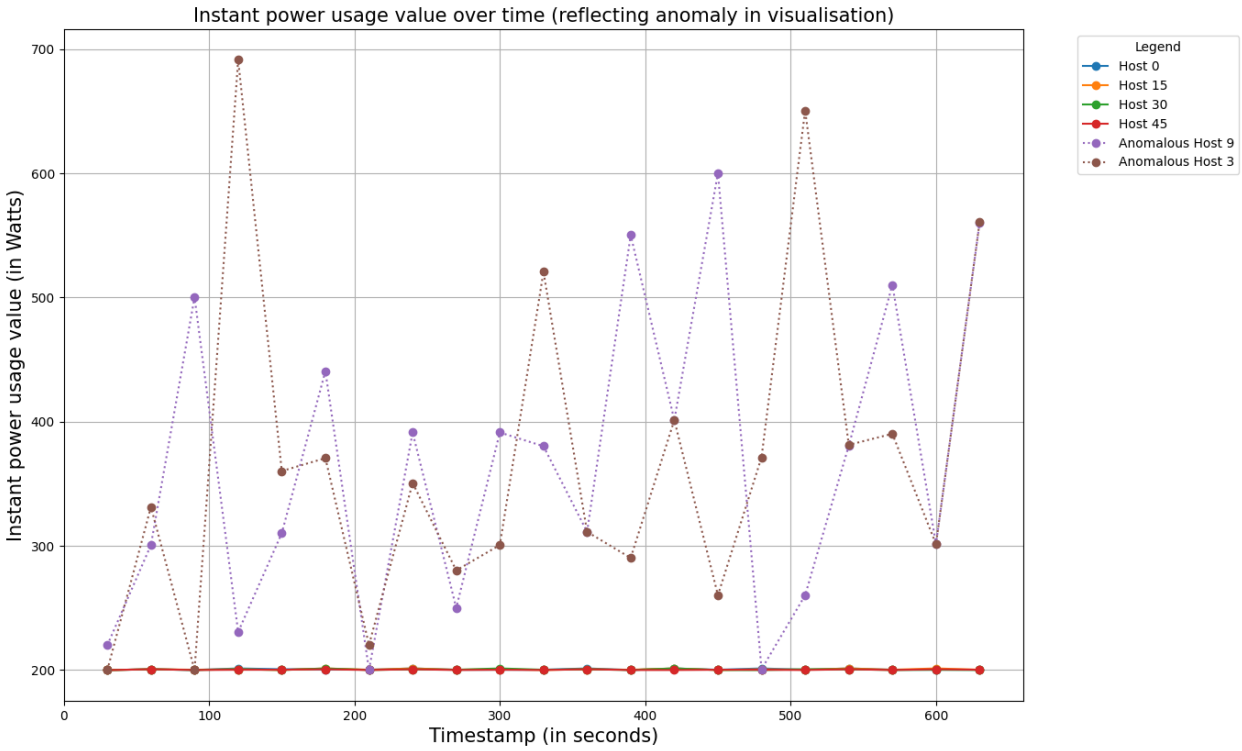


Figure 5.10: Node-based anomaly detection for abnormal power usage at different time intervals of trace execution with cluster-size equal to 150% of the standard DAS-6 cluster size

that would be the most efficient in terms of energy consumption. It is noticed that for a given configuration of a trace file and a cluster scale, the comparatively most efficient task scheduling algorithm is outplayed in another configuration with another trace file and the cluster scale. This clearly leads to the fact that task scheduling is an NP-hard problem, and there is no known best solution. The research community keeps advancing on optimal task scheduling techniques, which led to the introduction of the term "portfolio scheduling" (dynamic selection of a scheduling algorithm from a set of algorithms, or - the portfolio). This should be considered as part of future work in ODAbler, as it is out of the scope of this project.

Node-anomaly detection

This set of experiments conducted for node-anomaly detection reflects a nice result on how anomaly is detected based on a sensible threshold value chosen carefully based on the value used to fault inject on the power usage of multiple nodes. The assumption made for this experiment is that for an identical number of guests running on the hosts for a given duration, the power usage value of those hosts should be identical within that duration (because OpenDC is a discrete-event simulator). The actual results visualised in the respective Python notebooks reflect a deeper visualisation of anomaly and a detailed list of anomalous values for respective data points when the anomaly is flagged for the anomalous hosts.

The technique used in the current approach is not significantly advanced though, as independent work done in the field of anomaly detection involves sophisticated machine learning (ML) techniques these days. There are several independent research dedicated to anomaly

detection (discussed in Chapter 2 as part of background literature's related work section) that cover a lot of such techniques used in various fields like financial markets, data centre operations, etc. As part of the expansion of this 'ODAbler' project, the future plan of action is to design and implement some of the sophisticated ML models highlighting the capabilities of ODAbler as an advanced ODA framework acting as a strong digital twin for OpenDC.

5.5 Summary

In this section, we studied two experiments namely, (i) the energy-efficiency-based evaluation for multiple task-scheduling algorithms used for scheduling the workload tasks, and (ii) the node-based anomaly detection in terms of anomalous power-consumption value (originally injected via OpenDC on multiple nodes). We summarise the main findings obtained experimentally as listed below:

- **F1**. We noticed that there is no one policy among the available ones in OpenDC which is superior in terms of consuming less power. The results show that the power consumption varies with respect to the workload trace and the size of the host environment, and accordingly, one policy outperforms others in one experiment (and others in other experiments). This also hints at the introduction of "portfolio scheduling" (dynamic selection of a scheduling algorithm from a portfolio) as part of future work in ODAbler, which is out of the scope of this project. Please refer to Section 5.2 for more details about the current implementation.

- **F2**. In the nodes' anomaly-detection-related experiments, we inject anomalies in the power value of two fixed hosts in OpenDC dynamically at discrete intervals, and we try to detect the power-spike-related anomaly on all hosts. The ODAbler analyser is successfully able to detect the node anomaly on both hosts appropriately, and lists out the anomalous behaviour highlighting the timestamp of the node anomaly, which is very helpful for further analysis. Please refer to Section 5.3 for more details.

Chapter 6

Conclusion and Future Work

Enabling the operational data analytics (ODA) framework serves as a key driving factor in controlling the environmental footprint of the gigantic HPC, supercomputer, and other large-scale computing infrastructures. As the demand for computing resources increases, the need to regulate energy efficiency needs to be addressed, and performance-related questions arise equally. A holistic ODA framework can address them effectively, contributing to a greener society.
In this section, we will summarise the contributions, discuss some limitations which were encountered during the implementation of the project, and propose future opportunities in this field.

6.1 Conclusion

This thesis project proposes modelling an ODA framework called 'ODAbler' for a data centre simulator 'OpenDC', which allows easier experimentation for research purposes. The proposed capability aims to make it convenient and flexible for the researchers to conduct energy- and performance-related experiments, promoting several benefits which can be simulated based on a real-world data centre infrastructure.

- **RQ1: How to design the ontology of a large-scale computing infrastructure (typically HPC cluster) using the metrics exported in a time-series format?**
  In Chapter 3, we propose an exhaustive discussion on ontology modelling for an HPC cluster derived from the cluster metrics. We go through various requirements to be satisfied by the ontology, and then discuss the dataset description as reflected by the metrics. Lastly, we reuse an existing HPC to a significant extent wherever possible, and create two ontologies for SURF's LISA cluster and CINECA's Marconi 100 cluster based on their respective metrics. As the data source and exporters of both the collectors were different and heterogeneous (Prometheus export in SURF's LISA vs IPMI export in CINECA's Marconi 100), we modelled two distinct ontologies. However, these two could be merged to derive a full-fledged ontology (depending on the merge requirements)
- **RQ2: How to design an ODA framework for a data-centre simulator like OpenDC to realise some of the benefits of the ODA techniques?** In Chapter 4, we set the preface of a need for an ODA framework, and discuss the requirements. Further, we discuss the design elements of the ODAbler framework which encom-

passes OpenDC, Apache Kafka, InfluxDB + Telegraf, and ODAbler client analyser application. Additionally, we also discuss OpenDC in detail to support the community research on OpenDC in terms of simplified documentation about key classes and concepts. We discuss how changes are made in OpenDC to add the ODA module (which is a minor change not to impact the existing OpenDC functionality or affect the performance significantly), and the "digital twin" designed in the form of the 'ODAbler' client analyser application.

- **RQ3: What are the various (quantitative/qualitative) energy or performance benefits that have been realised as part of the experimental evaluation of the ODA framework?** This research question is addressed in Chapter 5, where it has been answered in the form of further sub-divided research questions: split into analysing the quantitative gains and qualitative gains respectively.
  - **How to validate the energy-awareness benefits realised (if any) after enabling an ODA framework for a large-scale computing infrastructure?** This research question is further subdivided into two questions aiming at the energy-efficiency (quantitative) and node-anomaly (qualitative) analysis capabilities of the ODAbler framework as a whole. Currently, the ODAbler framework is limited to these two quantitative and qualitative capabilities, which could be enriched with advanced analytical techniques and various other ODA capabilities as part of future work. In the section 5.2, we discuss how the ODAbler client analyser application analyses the energy usage values for each of the task-scheduling algorithms for a given workload trace file and a given cluster configuration, and highlights the most efficient task-scheduling algorithm among the three for that specific use-case. We see that there is no single winner in terms of the most energy-efficient algorithm, which could be attributed to the lack of energy-aware scheduling policy as part of our experimentation.
  - **How to validate the node-performance-related benefits realised (if any) after enabling an ODA framework for a large-scale computing infrastructure?** In the section 5.3, we discuss how the ODAbler client analyser application analyses the node-anomaly in power usage of hosts by comparing a standard non-anomalous result set with an anomaly-enabled result set. The ODAbler analyser is successfully able to visualise and highlight the timestamps of anomaly, based on our threshold value.

6.2 Limitations

The project has not been free of hiccups, to be honest. There were a lot of challenges in the implementation, and some of the implementation detail has certain limitations, as discussed below:

- The very first challenge in this work encountered was the dissimilarity of concepts between SQL-based relational databases and OWL2 language, which highlighted that it was pretty much infeasible to model the primary-key relationship for a time-series database. The concept of relational databases doesn't fit well in the graph-based ontology concept. As such, the modelled ontology has only one sample data record for each of the clusters. The addition of data in the graphical layout within the ontology is slightly challenging, and data stores or external APIs need to be integrated (in the form of a binding service).

- The choice of Kafka as a cluster to manage the messages has been done as part of familiarity with the message-brokering framework. At the same time, any other message brokering framework could have been selected and this option itself is open for research. The idea of selecting Kafka was driven because of its high popularity among the message brokering frameworks, as claimed by the Kafka team.

- The power-consumption model selected in OpenDC is a cubic power model based on CPU, for the sake of these experiments. However, after careful examination of the OpenDC simulator and the obtained results, it is clearly evident that the power model used (CPUPowerModel - cubic) is not errorproof, because of very high power usage results due to the introduced anomaly (which in itself is actually simple in nature that sets the power usage at discrete times of metric recording to a higher value of +10 Watts).

- The choice of scheduling algorithms has been done randomly. 2 of the task-scheduling algorithms (FIFO/STTO and Random) are already available in OpenDC, and the author extended it by adding HEFT as part of the exercise. The author also added a couple of others like MinMin and Ant-Colony optimisation (implemented during the distributed systems course which the author had taken long before), but it turned out that all of the externally added scheduling algorithms were reporting exactly the same power values at discrete intervals. In this case, it did not make sense to compare more scheduling policies, and thus it was decided to include only an additional algorithm not already available in the OpenDC repo originally: the HEFT algorithm. This could potentially be a bug to explore if those algorithms are also considered.

6.3 Future Work

Based on the limitations discussed in the previous section, there is a lot of scope for future research in the current project. In the future, the author aims to refine the integration between OpenDC and ODAbler. The author proposes some of the key areas which could be researched in advance, and also proposes some of the functionality-related improvements related to the current implementation, as outlined below:

- Improved user-friendly ODAbler interface for better visualisation (instead of the currently-used Python notebooks)

- Comparing advanced energy-aware scheduling (EAS) algorithms after their implementation in OpenDC instead of the current HEFT, Random and FIFO task-scheduling algorithms could showcase the advanced capabilities of ODAbler strongly.

- Extension to advanced analytical capabilities of ODAbler (including more ML-based analytics)

- Incorporation of more ODA functionalities (such as runtime tuning, scalability tuning, and workload modelling to name a few) in ODAbler

- Exploring other ways to contribute to improvement in the ODAbler framework: by adding value to the ODAbler framework (by tweaking either OpenDC configuration or ODAbler to generate more meaningful metrics, e.g., workload metrics).

Appendix A

Artefact Reproducibility

This chapter is dedicated to the knowledge base and additional metadata or other important pieces of information that could be interesting for the readers, and shares the artefacts which have been made available for experiments, or as part of the results. There are some available artefacts from the OWL ontologies in the form of graphical layouts, and then there are important information available for reproducibility (as part of the contribution to the FAIR initiative). The details are discussed in respective sections which are as follows.

A.1 Ontograf representation of OWL ontologies

This section shows the screenshot of an interactive visualisation of the relationships captured using Protégé desktop (or, Protégé), an environment for editing and managing ontologies [67, 68]. The version of Protégé used for plotting the Ontograf is version 6.5.1. OntoGraf[1] is a Protégé desktop plugin that provides support for visual, interactive navigation of the relationships in OWL ontologies. The steps to create an OntoGraf visualisation are listed below:

- Launch Protégé desktop application.
- Download any of the .owl ontology files representing the HPC ontology of an HPC cluster from the GitHub repo of "hpc-ontology-modeller" project[2].
- From the active Protégé window, select the "Direct imports" and upload any *.owl ontology file.
- Once the selected ontology is made active, navigate to the OWLViz pane for visualising the OWLViz format, or click on OntoGraf for visualising the OntoGraf format (as shown here). One can further click and expand the relevant classes for exploring further.

Fig. A.1 shows the OntoGraf visualisation of SURF's LISA cluster-driven OWL ontology, and Fig. A.2 shows the OntoGraf visualisation of CINECA's Marconi 100 cluster-driven OWL ontology.

[1] OntoGraf - `https://protegewiki.stanford.edu/wiki/OntoGraf`
[2] hpc-ontology-modeller - `https://github.com/am-i-helpful/hpc-ontology-modeller.git`

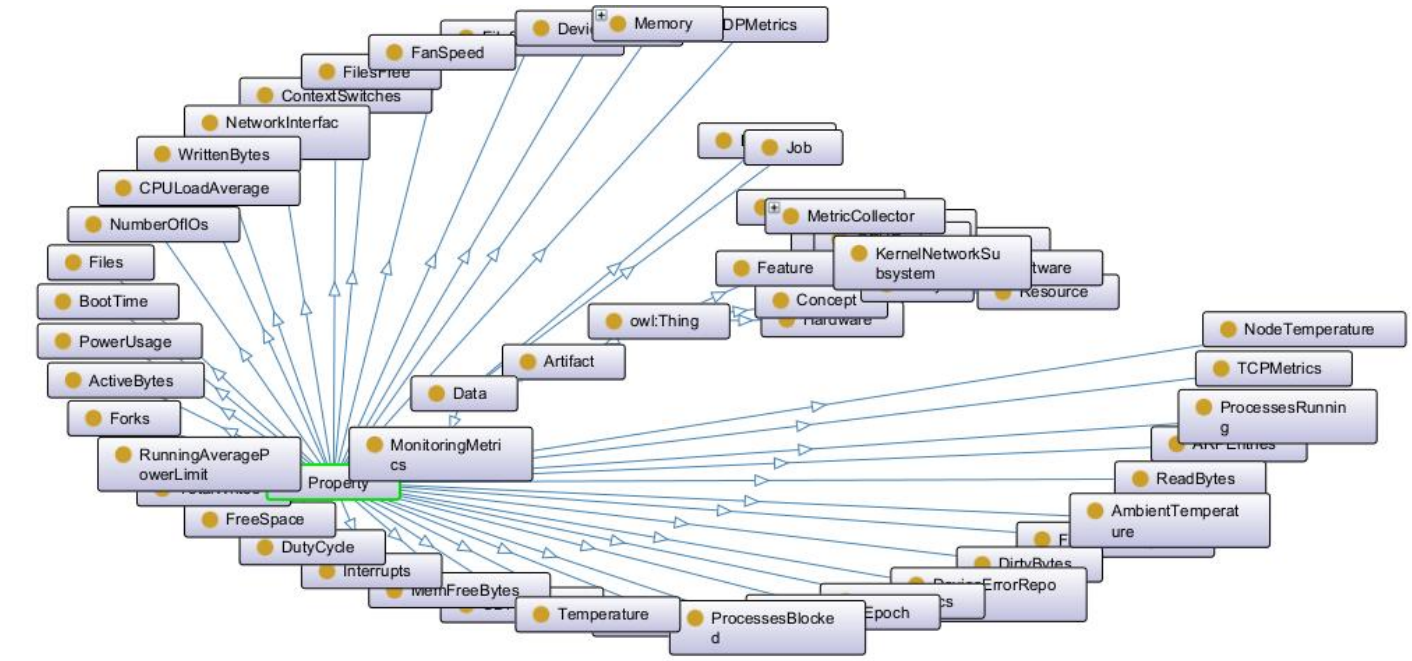


Figure A.1: SURF's OWL ontology OntoGraf visualisation (using Protege) showing the relationships.

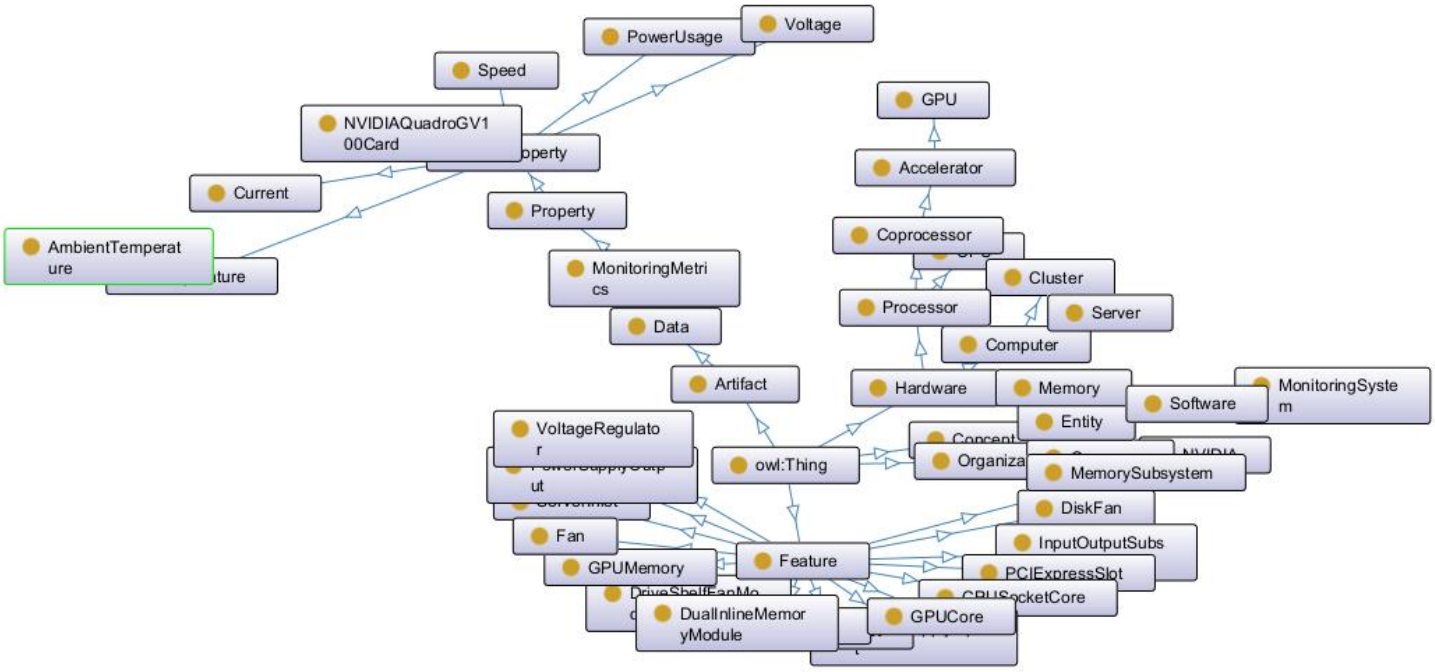


Figure A.2: CINECA's OWL ontology OntoGraf visualisation (using Protege) showing the relationships.

A.2 System setup for ODAbler-based experiments

These projects have been developed and tested on an HP laptop, model *"HP Pavilion 15-bc008tx"*, running Fedora 38 having kernel '6.4.15-200' at the time of experimentation. The system involves the OpenDC project available in IntelliJ IDE (version 2023.2), ODAbler Python-based project available in a Python-platform supported IDE (PyCharm in this case - version 2023.2), Apache Kafka application local setup (version 2.13-3.4.0), InfluxDB container (version 2.7.1), and Telegraf local agent installation (version 1.28.0). The detailed instructions for setting up OpenDC and ODAbler projects are available at the ODAbler GitHub repo. There are detailed video guides explaining the new functionalities too, inside the "documentation-guide" directory (in addition to the literature study document for an advanced overview of the ODA framework).

A.3 Projects' execution

The philosophy is critical to understanding what we mean by running experiments in a large-scale computing infrastructure within OpenDC and analysing in ODAbler. For using the OpenDC and ODAbler client (analyser) applications for executing the experiments for reproducibility, please follow the below instructions:

• Check out the respective project repositories to a location, and open them in IntelliJ and PyCharm IDE respectively (preferably).

• The main.py file contains the main declaration, where one can decide whether to launch an "energy-efficiency" analysis experiment or a "node-anomaly" detection experiment. It is suggested to only launch one experiment at a time, following the architectural flow. The default experimentation selected is the task scheduling algorithms' "energy-efficiency" analysis experiment.

• Make sure that Apache Kafka, and InfluxDB services are available and running on the system where the experiment is being conducted.

• Once both Kafka and InfluxDB are running, start the Telegraf agent with the configuration available in the key-configuration directory, under the filename telegraf.conf. It should be copied to the configuration directory located at /etc/telegraf/ on the system where the Telegraf service is to be started, such that Telegraf launches the service with expected results at the InfluxDB end.

• When all the pre-requisite services (Kafka, InfluxDB, and Telegraf) are running successfully, launch the OpenDC socket server (ODAExperimentListener) such that it can accept connection requests and experiment-related messages to kickstart the production of power and energy usage-related metrics, which can be shipped to Kafka, and then to InfluxDB via Telegraf. This process might take some time (2-5 minutes) depending on the experiment and the trace file executing at the moment.

• Please note that all sorts of configurations are automatically taken care of by the OpenDC server itself. ODAbler only needs to send either the experiment-type, and then wait for the response from OpenDC about the final status of the corresponding experiment.

• Once the metrics data is available at InfluxDB, the ODAbler client will start the analysis online (if there is no timeout). **IMPORTANT** - It is critical to note the time of starting the experiment (Flux uses UTC time standard to store the datetime; other time zones need to take care of the UTC conversion while querying in the on-demand mode of ODA analysis), as the same can be used later in Flux query for on-demand analysis (if needed). Otherwise, the options are to go through the InfluxDB "data explorer" and search for data points using the available filter, or install the InfluxDB tools (more details can be found here - https://awesome.influxdata.com/docs/part-1/introduction-to-influxdb-tools/).

• On-demand ODA analysis can always be done by triggering the analysis manually by invoking the opendc_energy_experiment_runner.py or opendc_anomaly_experiment_runner.py file inside the opendc_experiment_runner directory (and not using the main.py file).

• The runner modules call the already created template Python notebooks available inside the opendc_experiment_analyser directory, which has all desired notebooks ready for analysis within the respective experiment-named directory. Another important piece of information is that the template notebooks available inside the opendc_experiment_analyser directory are currently configured to search for a specific time duration (filtering on the start and stop time of the data available in InfluxDB, which is basically the timestamp of execution of the experiment; unless if the trace execution lasts longer writing a lot of data over a long range of time). More details about the time range can be read here - https://docs.influxdata.com/flux/v0/stdlib/universe/range/.

• **MOST IMPORTANT**: These template Python notebooks located in opendc_experiment_analyser directory, inside the directory /*experiment_name*/ - *.ipynb are copied to the generated_notebooks directory, and then executed for performing the desired ODA analysis. So, when the ODAbler has been set up for the first time, please make sure that the timestamp of the experiment execution is noted, converted to UTC format, and then the required changes are done to ensure that the data is filtered and analysed suitably.

• Lastly, please go through the individual Python notebooks to understand the analysis done, as they are self-explanatory and documented.

• Please use PyCharm IDE for setting up the ODAbler project and IntelliJ IDEA for the OpenDC project (if possible), as the author has not tested any other IDE/editor setup for executing the experiments.

Appendix B

Source Codes

B.1 HPC Ontology Modeller application

The source code for the "HPC Ontology modeller" application can be found on GitHub at the project-URL: hpc-ontology-modeller. This project consists of a README (documentation) file which describes the ontology modelling approach in detail.

B.2 ODAbler (ODA framework)

The ODA framework project consists of 2 components: the 'OpenDC' simulator (enabled with ODAbler design to export relevant monitoring metrics), and the corresponding ODA analyser engine - the 'ODAbler' client (analyser). Both together constitute what we call the ODAbler framework, which provides us with meaningful insights related to energy efficiency and anomaly identification in the trace execution in a given cluster environment. Both application's documentation (README file) is available on their respective GitHub pages respectively.

- The source code for the forked "OpenDC" application can be found on GitHub at the project-URL: opendc(forked).
- The source code for the "ODAbler" analyser application can be found on GitHub at the project-URL: ODAbler.

Bibliography

[1] N. Bourassa, W. Johnson, J. Broughton, D. M. Carter, S. Joy, R. Vitti, and P. Seto, "Operational data analytics: Optimizing the national energy research scientific computing center cooling systems," in *Proceedings of the 48th International Conference on Parallel Processing: Workshops*, pp. 1–7, 2019.

[2] F. Cappello, G. Al, W. Gropp, S. Kale, B. Kramer, and M. Snir, "Toward exascale resilience: 2014 update," *Supercomputing Frontiers and Innovations: an International Journal*, vol. 1, no. 1, pp. 5–28, 2014.

[3] O. Villa, D. R. Johnson, M. Oconnor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero, *et al.*, "Scaling the power wall: a path to exascale," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 830–841, IEEE, 2014.

[4] TOP500.org, "Green500 List - November 2022 | TOP500." `https://www.top500.org/lists/green500/list/2022/11/`, 2022. [Online; accessed 11-February-2023].

[5] A. Netti, D. Tafani, M. Ott, and M. Schulz, "Correlation-wise smoothing: Lightweight knowledge extraction for hpc monitoring data," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 2–12, IEEE, 2021.

[6] W. W. Eckerson, *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons, 2010.

[7] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, "Energy efficiency across programming languages: how do energy, time, and memory relate?," in *Proceedings of the 10th ACM SIGPLAN international conference on software language engineering*, pp. 256–267, 2017.

[8] A. Iosup, F. Kuipers, A. L. Varbanescu, P. Grosso, A. Trivedi, J. Rellermeyer, L. Wang, A. Uta, and F. Regazzoni, "Future computer systems and networking research in the netherlands: A manifesto," *arXiv preprint arXiv:2206.03259*, 2022.

[9] G. Andreadis, L. Versluis, F. Mastenbroek, and A. Iosup, "A reference architecture for datacenter scheduling: design, validation, and experiments," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 478–492, IEEE, 2018.

[10] J. Banks, *Discrete event system simulation*. Pearson Education India, 2005.

[11] F. Mastenbroek, G. Andreadis, S. Jounaid, W. Lai, J. Burley, J. Bosch, E. Van Eyk, L. Versluis, V. Van Beek, and A. Iosup, "Opendc 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters," in *2021 IEEE/ACM 21st Interna-*

tional Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 455–464, IEEE, 2021.

[12] A. Uta, A. Custura, D. Duplyakin, I. Jimenez, J. Rellermeyer, C. Maltzahn, R. Ricci, and A. Iosup, "Is big data performance reproducible in modern cloud networks?," in *17th USENIX symposium on networked systems design and implementation (NSDI 20)*, pp. 513–527, 2020.

[13] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision support systems*, vol. 145, p. 113524, 2021.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[15] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[16] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th international conference on E-science*, pp. 1–8, IEEE, 2012.

[17] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 446–452, IEEE, 2010.

[18] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[19] A. Iosup, O. Sonmez, and D. Epema, "Dgsim: Comparing grid resource management architectures through trace-based simulation," in *Euro-Par 2008–Parallel Processing: 14th International Euro-Par Conference, Las Palmas de Gran Canaria, Spain, August 26-29, 2008. Proceedings 14*, pp. 13–25, Springer, 2008.

[20] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "Groudsim: An event-based simulation framework for computational grids and clouds," in *Euro-Par 2010 Parallel Processing Workshops: HeteroPar, HPCC, HiBB, CoreGrid, UCHPC, HPCF, PROPER, CCPI, VHPC, Ischia, Italy, August 31–September 3, 2010, Revised Selected Papers 16*, pp. 305–313, Springer, 2011.

[21] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, pp. 185–209, 2012.

[22] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *2017 IFIP/IEEE symposium on integrated network and service management (IM)*, pp. 400–406, IEEE, 2017.

[23] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[24] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[25] D. Fernández-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.

[26] T. Wilde, A. Auweter, and H. Shoukourian, "The 4 pillar framework for energy efficient hpc data centers," *Computer Science-Research and Development*, vol. 29, pp. 241–251, 2014.

[27] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A conceptual framework for hpc operational data analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 596–603, IEEE, 2021.

[28] "Gartner it glossary." `https://www.gartner.com/en/information-technology/glossary`. Accessed: 2023-09-23.

[29] E. Bautista, M. Romanus, T. Davis, C. Whitney, and T. Kubaska, "Collecting, monitoring, and analyzing facility and systems data at the national energy research scientific computing center," in *Workshop Proceedings of the 48th International Conference on Parallel Processing*, pp. 1–9, 2019.

[30] E. Bautista, N. Sukhija, M. Romanus, T. Davis, and C. Whitney, "Omni at the edge," *Cybersecurity and High-Performance Computing Environments. Chapman and Hall/CRC*, pp. 63–84, 2022.

[31] A. Netti, M. Müller, C. Guillen, M. Ott, D. Tafani, G. Ozer, and M. Schulz, "Dcdb wintermute: Enabling online and holistic operational data analytics on hpc systems," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 101–112, 2020.

[32] A. Netti, M. Müller, A. Auweter, C. Guillen, M. Ott, D. Tafani, and M. Schulz, "From facility to application sensor data: modular, continuous and holistic monitoring with dcdb," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–27, 2019.

[33] D. Locke, "Mq telemetry transport (mqtt) v3.1 protocol specification," tech. rep., IBM, August 2010.

[34] R. Tracey, L. Hoang, F. Subelet, and V. Elisseev, "Ai-driven holistic approach to energy efficient hpc," in *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers 35*, pp. 267–279, Springer, 2020.

[35] C. Liao, P.-H. Lin, G. Verma, T. Vanderbruggen, M. Emani, Z. Nan, and X. Shen, "Hpc ontology: Towards a unified ontology for managing training datasets and ai models for high-performance computing," in *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pp. 69–80, IEEE, 2021.

[36] A. Iosup, L. Versluis, A. Trivedi, E. Van Eyk, L. Toader, V. Van Beek, G. Frascaria, A. Musaafir, and S. Talluri, "The atlarge vision on the design of distributed systems and ecosystems," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1765–1776, IEEE, 2019.

[37] G. G. Castañé, H. Xiong, D. Dong, and J. P. Morrison, "An ontology for heterogeneous resources management interoperability and hpc in the cloud," *Future Generation Computer Systems*, vol. 88, pp. 373–384, 2018.

[38] A. Zhou, K. Ren, X. Li, W. Zhang, and X. Ren, "Building quick resource index list using wordnet and high-performance computing resource ontology towards efficient resource discovery," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 885–892, IEEE, 2019.

[39] Y. Zhao, C. Liao, and X. Shen, "An infrastructure for hpc knowledge sharing and reuse," tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2017.

[40] A. Tenschert, "Ontology matching in a distributed environment," 2016.

[41] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *2008 Grid Computing Environments Workshop*, pp. 1–10, IEEE, 2008.

[42] F. T. Imam, "Application of ontologies in cloud computing: The state-of-the-art," *arXiv preprint arXiv:1610.02333*, 2016.

[43] W. Shin, V. Oles, A. M. Karimi, J. A. Ellis, and F. Wang, "Revealing power, energy and thermal dynamics of a 200pf pre-exascale supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, 2021.

[44] M. Terai, F. Shoji, T. Tsukamoto, and Y. Yamochi, "A study of operational impact on power usage effectiveness using facility metrics and server operation logs in the k computer," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 509–513, IEEE, 2020.

[45] A. Borghesi, M. Molan, M. Milano, and A. Bartolini, "Anomaly detection and anticipation in high performance computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 739–750, 2021.

[46] N. F. Noy, D. L. McGuinness, *et al.*, "Ontology development 101: A guide to creating your first ontology," 2001.

[47] L. Jean-Baptiste, "Ontologies with python," *Apress, Berkeley, CA*, 2021.

[48] J.-B. Lamy, "Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies," *Artificial intelligence in medicine*, vol. 80, pp. 11–28, 2017.

[49] C. A. Henson, H. Neuhaus, A. P. Sheth, K. Thirunarayan, and R. Buyya, "An ontological representation of time series observations on the semantic sensor web," 2009.

[50] A. Borghesi, C. Di Santi, M. Molan, M. S. Ardebili, A. Mauri, M. Guarrasi, D. Galetti, M. Cestari, F. Barchi, L. Benini, *et al.*, "M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer," *Scientific Data*, vol. 10, no. 1, p. 288, 2023.

[51] A. Borghesi, C. D. Santi, M. Molan, M. S. Ardebili, A. Mauri, M. Guarrasi, D. Galetti, M. Cestari, F. Barchi, L. Benini, F. Beneventi, and A. Bartolini, "M100 dataset 7: 22-04," Feb. 2023.

[52] S. Lohmann, V. Link, E. Marbach, and S. Negru, "Webvowl: Web-based visualization of ontologies," in *Knowledge Engineering and Knowledge Management: EKAW 2014 Satellite Events, VISUAL, EKM1, and ARCOE-Logic, Linköping, Sweden, November 24-28, 2014. Revised Selected Papers. 19*, pp. 154–158, Springer, 2015.

[53] M. Á. Rodríguez-García and R. Hoehndorf, "Inferring ontology graph structures using owl reasoning," *BMC bioinformatics*, vol. 19, pp. 1–9, 2018.

[54] J. Wu, F. Orlandi, D. O'Sullivan, and S. Dev, "An ontology model for climatic data analysis," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pp. 5739–5742, IEEE, 2021.

[55] R. Prodan, D. Kimovski, A. Bartolini, M. Cochez, A. Iosup, E. Kharlamov, J. Rožanec, L. Vasiliu, and A. L. Vărbănescu, "Towards extreme and sustainable graph processing for urgent societal challenges in europe," in *2022 IEEE Cloud Summit*, pp. 23–30, IEEE, 2022.

[56] A. Iosup, R. Prodan, A.-L. Varbanescu, S. Talluri, G. Magalhaes, K. Hokstam, H. Zwaan, V. Van Beek, R. Farahani, and D. Kimovski, "Graph greenifier: Towards sustainable and energy-aware massive graph processing in the computing continuum," in *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*, pp. 209–214, 2023.

[57] A. Netti, *Holistic and Portable Operational Data Analytics on Production HPC Systems*. PhD thesis, Technische Universität München, 2022.

[58] J. Kreps, N. Narkhede, J. Rao, *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, pp. 1–7, Athens, Greece, 2011.

[59] N. Garg, *Apache kafka*. Packt Publishing Birmingham, UK, 2013.

[60] S. N. Z. Naqvi, S. Yfantidou, and E. Zimányi, "Time series databases and influxdb," *Studienarbeit, Université Libre de Bruxelles*, vol. 12, 2017.

[61] N. Chan, "A resource utilization analytics platform using grafana and telegraf for the savio supercluster," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, pp. 1–6, 2019.

[62] P. Rattanatamrong, Y. Boonpalit, S. Suwanjinda, A. Mangmeesap, K. Subraties, V. Daneshmand, S. Smallen, and J. Haga, "Overhead study of telegraf as a real-time monitoring agent," in *2020 17th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 42–46, IEEE, 2020.

[63] C. Saad-Eddine and B. Younes, "Performance & energy consumption metrics of a data center according to the energy consumption models cubic, linear, square and square root," in *2019 7th Mediterranean Congress of Telecommunications (CMT)*, pp. 1–5, IEEE, 2019.

[64] S. Ma, A. Ilyushkin, A. Stegehuis, and A. Iosup, "Workflow trace archive shell trace," June 2019.

[65] P. Team, "Workflow trace archive pegasus_p2 trace," June 2019.

[66] L. Versluis, R. Mathá, S. Talluri, T. Hegeman, R. Prodan, E. Deelman, and A. Iosup, "The workflow trace archive: Open-access data from public and private computing infrastructures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2170–2184, 2020.

[67] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of protégé: an environment for knowledge-based systems development," *International Journal of Human-computer studies*, vol. 58, no. 1, pp. 89–123, 2003.

[68] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, and M. A. Musen, "Protégé-2000: an open-source ontology-development and knowledge-acquisition environment.," in *AMIA... annual symposium proceedings. AMIA Symposium*, pp. 953–953, 2003.