Energy-aware Information Modeling and Management for e-Infrastructures

This research received full funding from China Scholarship Council.

This work was carried out in the ASCI graduate school. ASCI dissertation series number XXX.

This research was also supported by the GreenClouds project (NWO), by the Dutch national program COMMIT and by the European Community Seventh Framework Programme under grant agreement no. 605243 (GN3plus).

Cover design by ...
Image courtesy by ...
Typeset by LaTeX
Printed and bound by ..
ISBN: XXX

Energy-aware Information Modeling and Management for e-Infrastructures

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. .....
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in Agnietenkapel
op dinsday 27 oktober 2015, te 12.00 uur

door

Hao Zhu

geboren te Jiangsu, China

I hear and I forget. I see and I remember. I do and I understand.

Xunzi

Contents

Contents i

Summary 1

Samenvatting in het Nederlands 3

1 Introduction 5

2 The Energy Description Language 15

3 The Energy Knowledge Base System 29

4 Evaluation of Power Estimation Models in a Computing Cluster 45

5 Joint Flow Routing-Scheduling for Energy Efficient Software Defined Data Center Networks 63

6 Conclusion 83

A OWL Schema of The Energy Description Language 87

B List of Abbreviations 97

C Source Code Repositories 99

List of Figures 101

List of Tables 103

General bibliography 105

Publications 113

Acknowledgements 115

Summary

e-Infrastructures consisting of data centers, networks and collaborative environments are a cost-effective solution for hosting Cloud and Grid applications for research purpose. However infrastructures incur tremendous energy costs and CO2 emissions. Energy management focuses on technologies for efficiently scheduling applications and allocating resources in a distributed infrastructure with energy as an important factor in the policy and cost evaluations. For example, consider consolidating the applications onto parts of servers and switching idle servers into low-power mode. Different from commercial data center owners like Google, where each data center is usually independent and energy monitoring and management tools only work in the local domain, scientific e-infrastructures run across multiple administrative domains. Here the information on the footprint needs to be exchanged and energy management techniques need to span across multiple domains. Energy management in e-infrastructures relies on knowledge of the energy footprint for different classes of distributed applications and the configuration and structure of the equipment across different service providers. Therefore, a distributed energy-information system is needed to organize and provide the knowledge for energy management.

The knowledge in the information system should be carefully monitored and organized, as incomplete or badly-organized information hinders optimal decision-making during energy management; the incoherent information impedes the information exchange for the energy management across multiple administrative domains. We set out to define a semantic information model, which represents concepts and the relationships for capturing the knowledge using the Semantic Web, for the information system. The Semantic Web provides an effective mechanism for data interoperability and knowledge sharing. With the energy knowledge of infrastructures, we then aimed to design energy management strategies for executing applications in an energy efficient manner.

Based on the introduction above, my work answered two research questions discussed in this thesis: 1) What is the proper approach to design and create an energy-aware information model for the description of e-Infrastructures and develop a sufficient information system for their energy monitoring? 2) What new energy management techniques will emerge by applying the developed information model and knowledge base system?

Although my research focuses on e-infrastructures, the outcomes can be extended to generic data centers. The scientific contributions presented in the thesis are as follows:

1. We create the Energy Description Language (EDL), which is a semantic information model for the description of e-infrastructures with energy-awareness. The EDL ontology reuses the Infrastructure and Network Description Language (INDL) to describe the resources and network infrastructure that connects these resources.
2. DAS-4 is a distributed e-infrastructure used by universities and organizations in the Netherlands for the purpose of research and education. We build the Energy Knowledge Base (EKB) system for energy monitoring in DAS-4. As far as we know, EKB is the first implementation of a semantic-based energy-aware information system, which leverages EDL to model dynamic energy-related states of resources in the computing and network layers.
3. Power consumption of a server is a function of states of resource components that can be obtained from the Operating System (OS) or Performance Monitoring Counters (PMCs). We design and create a set of non-linear approaches to estimate the power consumption of servers. Based on measurement data from DAS-4, we evaluate the accuracy, portability and usability of the linear and non-linear approaches. Our work shows the multiple-variable linear regression approach is more precise than the CPU only linear approach. The neural network approaches have a slight advantage – its root mean square error is at most 15% less than that of the multiple-variable linear approach. But the neural network models have worse portability when these models are applied to homogeneous nodes across the same or other clusters. The Gaussian Mixture Model has the highest accuracy on nodes but requires the longest training time.
4. OpenNaaS is a management platform that enables the abstraction of underlying network technologies and offers NaaS-based services. We implement an efficient framework for green routing in data center networks based on OpenNaaS. The energy-aware OpenNaaS uses EDL as the information model for the energy-aware monitoring and description capabilities. We also study the design and selection of energy-aware routing strategies for the prototype. The optimized strategies combine flow routing algorithms that make routing decisions for the flows and flow scheduling algorithms that schedule the flows on the same link. Different from previous power-minimization studies, we evaluate the energy consumption of the strategies. Our simulation shows that the combination of priority-based shortest routing and exclusive flow scheduling has higher energy efficiency without performance degradation, particularly when traffic consists of large-sized flows.

Samenvatting

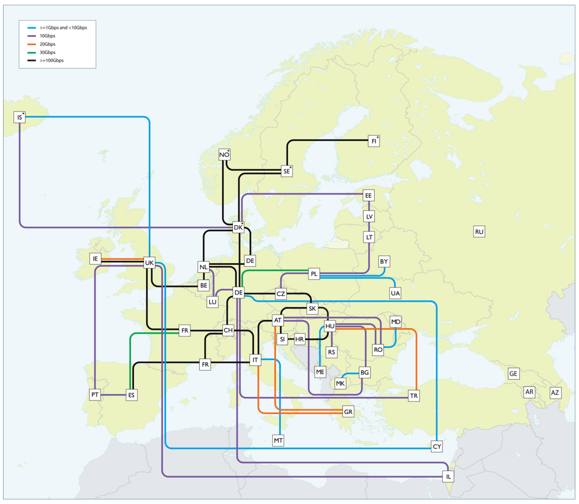
Chapter 1

Introduction

1.1 e-Infrastructures

e-Infrastructures consist of networks, data centers and collaborative environments. They facilitate scientific research carried out through global collaborations across regions. They are mainly used for scientific data processing.

For example, GÉANT [3] is the pan-European research and education network that interconnects Europe's National Research and Education Networks (NRENs) and other scientific infrastructures e.g. LOFAR [34] and SKA [35]. Altogether GÉANT connects the servers or data centers of over 10,000 institutions, providing the services in the area of networking, data center computing and storage. Its European connections are shown in Fig. 1.1. Another example is DAS-4 [8], an experimental e-Infrastructure built for computer science researchers who work on various aspects of parallel and cloud computing and large-scale multimedia content analysis in different regions of the Netherlands.



GÉANT connectivity as at January 2014.

Figure 1.1: GÉANT European connections [19]

In today's big data era, large amounts of scientific data in the meteorology, genomic, and biological and environmental research domains are being produced. e-Infrastructures are being developed world-wide to meet the soaring demand for computing and transport of scientific data. For example, in SKA approximately 160 Gigabits (10^9 bits) per second of data will be transmitted from each radio dish to a central processor; this means that the dishes will produce at least ten times the current global internet traffic [35]. SKA will require an unprecedented mount of high speed networks and super-computer scale data centers.

The increasing size and utilization of e-Infrastructures will lead to a large amount of energy consumed and lots of greenhouse gas (GHG) emissions that will contribute to making global warming worse. In fact, recent energy statistics indicate that the entire data center industry produces 1.5-2% of global energy consumption [102]. The predictions for annual increases in data center power demand are as high as 15-20% [48]. According to the U.S. Environmental Protection Agency (EPA), every 1000 kWh of energy consumption due to the Information and Communication Technology (ICT) industry leads to 0.72 tons of CO2 emissions, which is even more than the volumn of emissions per 1000 kWh created by vehicles [31]. Even network participants in GÉANT are now considering about their environmental impact. For example, a study by GRNET (Greek Research and Technology Network) showed that their core networks and data centers produced the equivalent of over 7509 tons of CO2 in 2010 [132].

The statistics above highlight the need for understanding and controlling energy consumption and GHG emissions in e-Infrastructures. Commercial data center owners (such as Google, Apple and Microsoft [86]) have started monitoring their energy consumption and employing various technologies to lower their operating cost as well as reduce their environmental impact. For instance, Google has the most energy efficient data centers in the world [22].

On the contrary, e-Infrastructures are still being operated without energy information on their footprint, and lag behind in supporting energy management. However, adapting energy monitoring and management tools from commercial data centers cannot be adopted as is in e-Infrastructures. Each commercial data center is usually independent and energy monitoring and management tools only work in the local domain; a scientific e-Infrastructure runs across multiple administrative domains, where the information on the footprint needs to be exchanged and energy management techniques need to span across multiple domains.

In this thesis, we research energy monitoring and energy management suitable for e-Infrastructures. We propose a novel semantic approach for describing e-Infrastructures and their energy-related states, and for organizing energy knowledge in our studies of energy monitoring. Leveraging our semantic models and our energy monitoring framework, we investigate new algorithms for estimation of power consumption in servers and energy-aware routing for networks.

1.2 The Statement and Goals of Energy Management

Energy management technologies focus on the efficient scheduling of workloads and on controlling resources, using energy and GHG emissions as important factors in policy and cost evaluation.

In order to clearly define the important concepts involved in this thesis, we describe the components of ***energy management*** in an infrastructure as shown in Fig. 1.2. We identify three layers:

- Application domain;
- Control domain;
- Infrastructure layer.

The control domain includes the actual information system and energy management system, and it interacts with the application domain and the infrastructure. The information system monitors and provides runtime information about the infrastructure such as load and power consumption (or CO2 emissions) of resources when running previous workloads; network topology; and available resources. The energy management system contains an optimizer and a scheduler. When the information on the workloads, Service Level Agreements (SLAs) and the infrastructure is available, the optimizer can decide on an energy-aware assignment of workloads to resources that satisfies the SLAs. The scheduler carries out scheduling activities according to this decision; it assigns workloads and controls the states of resources, e.g. switching resources on/off, via commands or scripts. The energy management approach about workload scheduling we present here is called ***energy-aware workload scheduling***.

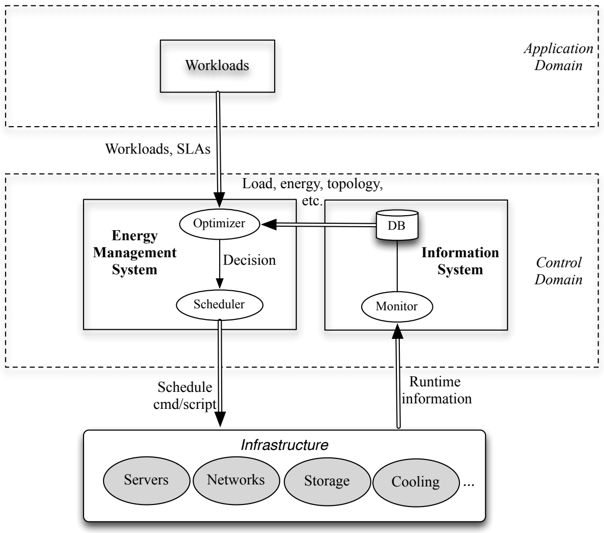


Figure 1.2: The components of energy management in infrastructures.

Apart from energy-aware workload scheduling, there are two other possible types of energy management [86]. The first is ***energy-aware virtual machine (VM) management***, which focuses on a virtualized environment in data centers [33]. In this case, mapping workloads to VMs is achieved by VM migration. This is similar to workload scheduling.

The second type is ***power capping management***, a method to control peak power consumption [89] [129]. Servers in data centers have been traditionally designed with over-provisioned cooling and power delivery systems. In reality, most servers don't come close to reaching maximum capacity. Such over-provisioning adds cost to the system and enlarges the server footprint, but benefits few real workloads. Power capping limits the amount of power by managing states of hardware resources to allow cooling and power delivery system to be under-provisioned and maintain safety at all times. It is not suitable for e-Infrastructures as scientific computing and simulation in them often use resources at their peak performance. Power capping management is beyond the scope of this thesis.

The ideal goals that energy management techniques in a data center should achieve are described:

- Power-proportionality. Data centers consume power in proportion to the amount of work performed. Data centers should consume no power when idle, gradually more power as the computing load increases, and maximum power as all the equipment acts at peak performance. Power-proportional data centers can run jobs without redundant power distribution. Fig. 1.3 shows that power consumption of a typical server is not proportional. Most of network equipment is not power proportional either [38] [97].

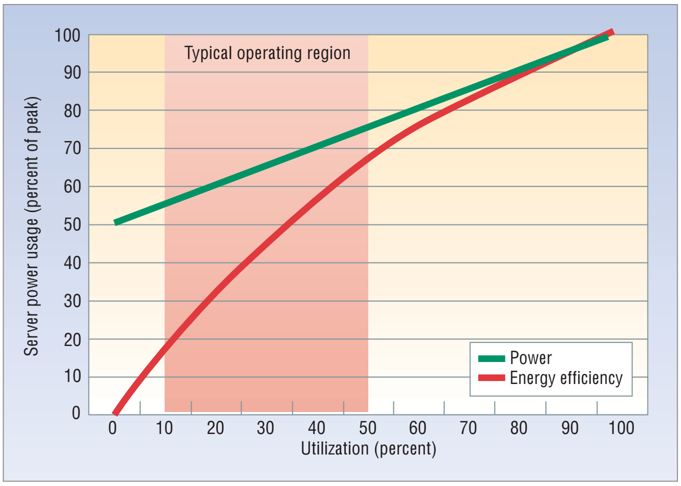
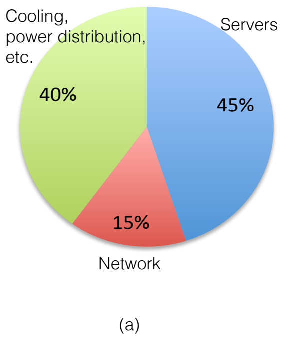


Figure 1.3: Server power usage and energy efficiency at varying utilization levels, from idle to peak performance [44].

- Power Usage Effectiveness (PUE) close to 1. A data center is a facility housing a large group of networked servers, associated power distribution equipment and cooling facilities. In order to measure how efficiently a computer data center uses energy, PUE is defined as the ratio of total data center power usage to IT equipment power usage. If PUE is close to 1, then data centers spend all the power supplies on the IT equipment for useful computing and communication work. Fig 1.4 (a) provides a rough guide to associated electricity costs in a data center. The average PUE value of commercial data centers was 2 in 2005 [88]. There is lots of progress in recent years. PUE for Google data centers has dropped to 1.12 in the third quarter of 2014 [22]. PUE for the supercomputer Cartesius, which is part of an e-infrastructure offered by SURFsara [36], is 1.2 in early 2014 [95].

- Carbon Usage Effectiveness (CUE) close to 0. CUE is defined by the ratio of total CO2 emissions caused by the total data center energy usage to IT equipment energy usage. CUE in ideal data centers should be close to 0. Data centers should effectively use clean energy sources and have no carbon emissions.



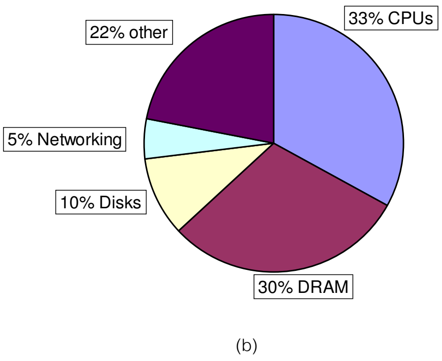


Figure 1.4: (a) Costs distribution in a data center [72]; (b) Server peak power by hardware components from a Google data center [79].

PUE and CUE are the metrics to measure the effect of energy management techniques. Energy efficiency is also a common energy metric. A technique is energy efficient if it has higher performance when consuming the same energy or it has the same performance when less energy is used.

1.3 Energy-aware Information Modeling for e-Infrastructures

As we discussed, workload scheduling is the energy management strategy we adopted in our work. For instance, scheduling latency-tolerant workloads onto nodes with green energy sources or consolidating workloads into a smaller set of nodes provide effective mechanisms for reduction of energy costs and GHG emissions. But applying all these energy management approaches requires energy-related knowledge of the infrastructures.

In the case of an e-Infrastructure, which consists of distributed administrative domains, the operators have the opportunity of cooperating with each other to perform global power management. We have already seen some attempts in NRENs. For example, NRENs in Europe cooperated to develop a green routing technique across different networks to transfer data in the Mantychore Project [17]. To support these cases, an information system for an e-Infrastructure is needed to organize and provide the energy-related knowledge, which should include energy footprint for different classes of distributed applications and configuration of resources in different administrative domains. This system maintains information in each local domain and provides the information to peers in other domains to form knowledge of the overall e-Infrastructure.

The knowledge in this information system should be carefully monitored and organized, as incomplete and badly-organized information hinders optimal decision-making during energy management; data centers in an e-Infrastructure have usually heterogeneous hardware and software components, so incoherent information impedes information interoperability for energy management across multiple domains.

To alleviate this problem, we introduced an energy-aware information model to describe concepts and the relationships of e-Infrastructures for capturing the knowledge in the information system. The information model is used to describe managed objects at a conceptual level, independent of any specific implementation used to process data. The operators can use the models to know how to properly manipulate data in an information system.

There is an additional benefit of introducing an energy-aware information model. The state of the art in the Green ICT area lacks a deep understanding of the complicated infrastructure components and their states, their correlations and inter-dependencies: for instance, which states and properties resources have, which are useful for the power management, and how the states can be measured. This obscurity hinders the progress of Green ICT. Using models to describe the infrastructure with energy-awareness can help the research community achieve a better understanding of the available resources for designing energy management technologies.

1.4 Energy Management for e-Infrastructures

We also investigate energy management for e-Infrastructures in this thesis. The potential of saving power from power distribution and cooling equipment in e-Infrastructures is pretty limited. Currently, the scale of an individual data center in e-Infrastructures is not big enough compared to a commercial data center which usually runs ten thousands or millions servers [29], so it is not cost-effective to buy and install the enhanced auxiliary equipment, e.g. a high-voltage power supply, or water cooling system. Moreover, the power supply and cooling infrastructure is fixed, unable to be optimized or replaced. The average PUE of some e-Infrastructures is not too high, for example the PUE of a DAS-4 cluster in University of Amsterdam is estimated at about 1.4. Therefore, we mainly focus on the techniques of managing energy consumption of the IT equipment in e-Infrastructures in this thesis.

When an energy management system assigns workloads onto the servers of e-Infrastructures, the most important factors are estimation of the incoming workloads as well as estimation of the energy consumption of servers as a result of the workload schedule to be carried out. Considering that energy consumption equals to power consumption multiplied by time, it is necessary to estimate power consumption[1].

A typical server includes several hardware components: CPU, memory, disks, networking devices, etc. The proportion of power consumption on hardware components in a typical server is shown in Fig. 1.4 (b). The majority of power estimation models used before are linear; they are linear functions of the load of these components [42][134]. But some previous work found that the power consumption of a server is not fully linear with CPU or I/O load [62][52][90]. This motivates us to build non-linear mathematical models suitable for power estimation and evaluate their effect.

In the total energy footprint of data centers, networks only account for about 15% of the total expense (See Fig. 1.4 (a)), and they are not the largest cost category. Thus, the network component has rarely been considered the most relevant component for energy optimization. However, the proportion of networks could grow to 50% in a data center where power management techniques are only used on the server-side [38]. This occurs particularly in data centers in an e-Infrastructure where large volumes of data are frequently transported. It's therefore crucial to reduce the energy consumption of data center networks in e-Infrastructures.

Energy-aware routing techniques are effective to save energy by making routing decisions to aggregate traffic over a subset of links and devices in over-provisioned networks and switch off unused network components. Some energy-aware routing studies are theoretical but they neglected the scheduling algorithm for routed flows on the same link [136] [131]. Some studies provided actual implementations and prototypes, but they only have limited applicability [78] [124]. This motivates us to propose a more useful solution for energy-aware routing and investigate a joint routing-scheduling algorithm.

1.5 Research Questions

Current information systems organize and provide accurate information for resource management in e-Infrastructures, e.g. resource type, resource state, and network topology. For instance, the Monitoring and Discovery System [116], which is developed for Grid infrastructures like EGEE [10] and its successor EGI [2], collects and aggregates information about resources and services. However, these information systems cannot support energy management due to lack of the capabilities of monitoring energy information.

We aim to study and build an energy-aware information system. Before that, we have to look into an energy-aware information model to describe e-Infrastructures with energy awareness. Thus, our first research question can be described as:

Q1: What is the proper approach to design and create an energy-aware information model for the description of e-Infrastructures and develop

[1]We differentiate the concepts of *power consumption (Watt)* and *energy consumption (Joule)* in this thesis.

a sufficient information system for their energy monitoring?

Once we build an energy-aware information model and information system, we have the basis for studying energy management techniques for e-Infrastructures. Our second research question can be naturally stated as:

Q2: What new energy management techniques will emerge by applying the developed information model and information system?

From the information system we developed for an e-Infrastructure, we can obtain the measurement data in terms of performance of hardware components and power consumption of servers. We aim to derive power estimation models using different non-linear approaches. Thus, a more detailed question that is part of question Q2 follows:

Q2a: How do we build and evaluate non-linear approaches for power estimation in a cluster environment?

We aim to explore an energy-aware component in an open-sourced platform to implement energy-aware routing for data center networks. With the energy-aware information model, we can design data elements and their structure of energy monitoring for networks. We also intend to evaluate the effect of different energy-aware routing strategies and select a optimal one for this component. Thus, the second detail question in Q2 is proposed as:

Q2b: What is a proper way to explore energy-aware routing in data center networks and how much impact do routing strategies have on the energy efficiency of the networks?

1.6 Contributions and Thesis Outline

According to the research questions proposed, the thesis is structured as follows. Chapter 2 and Chapter 3 investigate the answer of the question *Q1*. Chapter 4 and Chapter 5 study the answer to the questions *Q2a* and *Q2b* respectively.

- **Chapter 2** reviews the basic idea of the Semantic Web, and gives the motivations for using a semantic approach to describe e-Infrastructures with energy-awareness. The description of e-Infrastructures with energy-awareness require two models. This chapter first presents our group's previous work on the semantic model for computing and network infrastructures – the Infrastructure and Network Description Language (INDL) [71]. Then this chapter presents the Energy Description Language (EDL) [138] model for energy monitoring and how we build EDL upon the Infrastructure and Network Description Language.
  In this chapter our main contribution is Energy Description Language. EDL is OWL-based model, which improves the interoperability of energy knowledge across multiple domains. The concepts in EDL can support a wide range of power management scenarios such as power estimation and green resource discovery.

- **Chapter 3** validates the Energy Description Language ontology by using its concepts in the Energy Knowledge Base (EKB) [137] system that is a distributed information system for energy monitoring in e-Infrastructures. This chapter discusses the architecture of EKB, and presents the usecases and evaluation of EKB.
  We build the Energy Knowledge Base system for energy monitoring in DAS-4. As far as we know, EKB is the first implementation of a semantic and energy-aware information system, which leverages EDL to model infrastructures with energy awareness. EKB can monitor and provide the knowledge for energy-aware resource allocation as well as resource discovery.
- **Chapter 4** studies different linear and non-linear approaches to estimate power consumption of an entire server. We summarize our contributions of the chapter as follows:
  - we design and create a set of non-linear approaches to estimate power consumption of servers;
  - we build and evaluate the power estimation models in a cluster environment using a large amount of measurement data;
  - we evaluate the accuracy, portability and usability of the linear and non-linear approaches.

  Our work shows the multiple-variable linear regression approach is more precise than the CPU-only linear approach. The neural network approaches have a slight advantage – its root mean square error is at most 15% less than that of the multiple-variable linear approach. But the neural network models have worse portability when the models generated on a node are applied across its homogeneous nodes. The Gaussian Mixture Model has the highest accuracy on Hadoop nodes but requires the longest training time.
- **Chapter 5** presents a prototype of energy-aware OpenNaaS, a solution to energy-aware routing in a network management platform. This chapter also discusses the design and selection of energy-aware routing strategies for the prototype.
  The detailed contributions of the chapter are the following.
  - We implement an efficient framework for green routing in data center networks based on OpenNaaS. OpenNaaS is an open-sourced management platform that enables the abstraction of underlying network technologies and offers NaaS-based services.
  - We study the design and selection of energy-aware routing strategies for the prototype. The optimized strategies combine flow routing algorithms that make routing decisions for the flows and flow scheduling algorithms that schedule the flows on the same link. Different from previous power-minimization studies, we evaluate the energy consumption of the strategies. Our simulation shows that the combination of priority-based shortest routing and exclusive flow scheduling has higher energy efficiency without performance degradation, particularly when the size of flows is large.

Chapter 6 summarizes the overall research contributions in this thesis, and discusses the conclusions to the research questions.

Chapter 2

The Energy Description Language

—A semantic approach for modeling infrastructures with energy awareness

This chapter is based on:

• M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, **H. Zhu**, Z. Zhao and C. de Laat (2013). A semantic-web approach for modeling computing infrastructures. Computers and Electrical Engineering, 39 (8), 2553-2565.
• **H. Zhu**, K. van der Veldt, P. Grosso and C. de Laat (2014). EDL: an energy-aware semantic model for large-scale infrastructures. (Technical report UVA-SNE, no. 2014-02). Amsterdam: Universiteit van Amsterdam, System and Network Engineering. `http://dare.uva.nl/document/2/161769`
• **H. Zhu**, K. van der Veldt, P. Grosso, Z. Zhao, X. Liao and C. de Laat (2012). Energy-aware semantic modeling in large scale infrastructures. In Work in Progress Sessions (WiP): 2012 IEEE International Conference on Green Computing and Communications (GreenCom 2012): pages 11-14, Besancon, France.

2.1 Introduction

In Sec. 1.3 of the previous chapter, we showed the use cases of workload scheduling. They require the description of e-infrastructures with energy-awareness to capture energy-related knowledge. In Sec. 2.2 of this chapter, we describe the Semantic Web, and explain why such a semantic approach is suitable for describing e-infrastructures with energy-awareness .

The energy-related knowledge of an e-infrastructure includes the configuration and structure of resources and the energy footprint of applications. Correspondingly, we will require two models: one for describing the structure and capabilities of the infrastructure and another for describing the energy-related states of the infrastructure.

We first provide a brief introduction to our previous work on the semantic modeling of computing and network infrastructures – namely the Infrastructure and Network Description Language (INDL) in Sec. 2.3.

Then we present our Energy Description Language model and explain how it imports INDL to describe infrastructures with energy-awareness in Sec. 2.4.

EDL itself focuses on the concepts and relationships of energy monitoring and measurement, which are used for energy management. Sec. 2.4.1 presents the state of the art on the energy-aware information models. Sec. 2.4.2 describes the EDL model and its components. Sec. 2.4.3 discusses the features of EDL.

2.2 Semantic Web Framework

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [9]. It was first proposed as a way for machines to understand web pages and data. To further understand the power of Semantic Web, we take a simple example. Consider the following two statements:

- A links to B.
- There is a link between A and B.

If a computer receives this example information, it won't understand that these two statements mean the same thing. But the Semantic Web can represent two sentences with the same semantics using a Resource Description Framework (RDF) triple: *{A, linkTo, B}*. This is where the Semantic Web can help computers to parse information and derive knowledge.

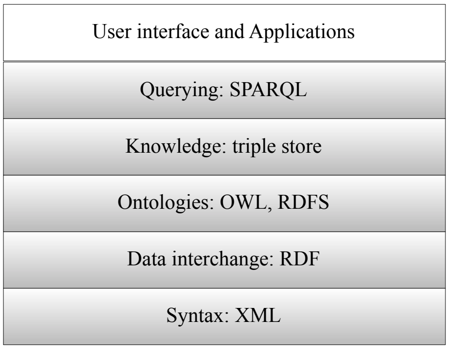


Figure 2.1: A simplified architecture of the Semantic Web

Fig. 2.1 illustrates the simplified architecture of the Semantic Web derived from [80]. The functions and relationships of the components can be summarized as follows:

- XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within. N3 [13] and Turtle [14] are similar syntaxes.
- RDF [60] is a simple language for expressing data resources, in particular for representing metadata about data resources, which refer to resources and their relationships. RDF provides a standard model for data interchange on the Web. RDF is a fundamental standard of the Semantic Web.

- RDF Schema (RDFS) [7] and OWL [59] are languages for describing ontologies. Ontologies are information models in the Semantic Web. Both RDF Schema and OWL extend RDF. An RDF-based ontology can be represented in a variety of syntaxes, e.g. RDF/XML, N3 and Turtle. One of the most popular syntaxes for OWL is RDF/XML which uses an XML syntax to describe RDF triples.
- Knowledge is structured data in the format of RDF triples, which is saved in a unique type of database called triple store, examples of which includes Sesame, Jena DB and AllegroGraph.
- SPARQL [111] is a protocol and query language for knowledge. Applications retrieve information from a triple store using SPARQL.

2.2.1 Resource Description Framework and ontology description language

In the Semantic Web, data is expressed as individuals or instances. They can be described by a set of RDF triples, namely an RDF graph. Each triple has a format of *{Subject, Predicate, Object}*:

- *Subject* represents the resource to be described.
- *Object* represents another resource or the value of the property for the subject.
- *Predicate* is a property relevant to the subject; it indicates the relationship between *subject* and *object.*

Fig. 2.2 is an example of an RDF graph, which contains a number of triples. Here we see for example one is "{HadoopNode01, inDomain, UvA}". The graph also contains other triples representing the IP information and the type of "HadoopNode01" and "HadoopNode02".

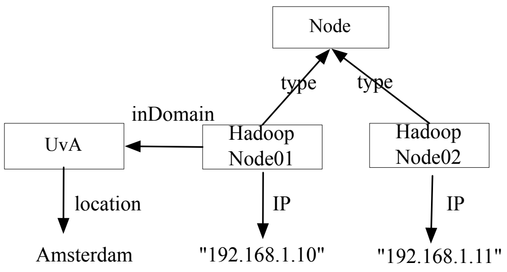


Figure 2.2: A simple example of RDF graph

RDF provides a common framework for expressing data so it can be exchanged between applications without loss of meaning. RDF uses Uniform Resource Identifiers (URIs), which are universal global unique identifiers, to identify resources or properties. URIs can ensure the uniqueness of data.

RDF Schema extends RDF and is a vocabulary for describing properties and classes of RDF resources. Same type of resources belong to a class. RDF Schema provides mechanisms for describing groups of related resources and the relationships between these resources. For example, the property "rdf:type" is used to specify the certain type of resource. In Fig. 2.2, "HadoopNode01" and "HadoopNode02" are the same type of resources, and they are all computing "Nodes". RDF Schema introduces the *domain* of and the *range* of a property to define what kind of types are valid as subject and object. The set of valid subjects is called the domain; the set of valid objects is called the range of that property. OWL further extends the vocabulary of RDF Schema in terms of the relations between classes (e.g. Man and Woman can be stated to be *disjoint* classes), equality, richer typing of properties, characteristics of properties, and enumerated classes.

2.2.2 Benefits of Semantic Web

We believe that there are several advantages to using a semantic approach to describe e-Infrastructures with energy-awareness. The major ones are:

- Data is interoperable. The data of each domain in an e-Infrastructure should be interoperable such that it can be understood and shared between each domain. RDF is a common framework that supports this data interoperability in the multi-domain case. RDF links knowledge from distributed domains using URIs and it allows to combine the knowledge.
- Ontologies are extensible. OWL defines classes, properties and how they can be imported in an ontology, so that an OWL ontology can use the vocabulary of other ontologies. OWL provides explicit separation between semantics and syntax. The clear separation also helps the ontology developers to mix different ontologies.

Apart from Semantic Web, other possible approaches might rely on XML Schema [32]. XML Schema is a language that describes and restricts the structure and content of elements in an XML document. XML Schema can also define information models. XML models are more concise than OWL ontologies in XML/RDF. But when machines consume information, this is not a real concern. It deserves mentioning that OWL ontologies can be described using other more compact syntaxes, such as N3 to keep concise.

But XML Schema presents several disadvantages. XML models are not extensible. For instance, if an element in a information model needs to be extended, *e.g.* adding a new attribute, all its descendants of this model have to be updated to include this new attribute. OWL ontologies are extensible as they do not restrictively define the structure of a document. In addition, XML lacks the restrictions on identifiers, and it is not straightforward to create globally unique identifiers across multiple domains in the XML models.

2.3 Infrastructure and Network Description Language

The goal of INDL is to capture the concept of computing and network infrastructures and to describe the storage and computing capabilities of the resources. The INDL ontology is built upon the Network Markup Language (NML) ontology. Both are OWL-based ontologies.

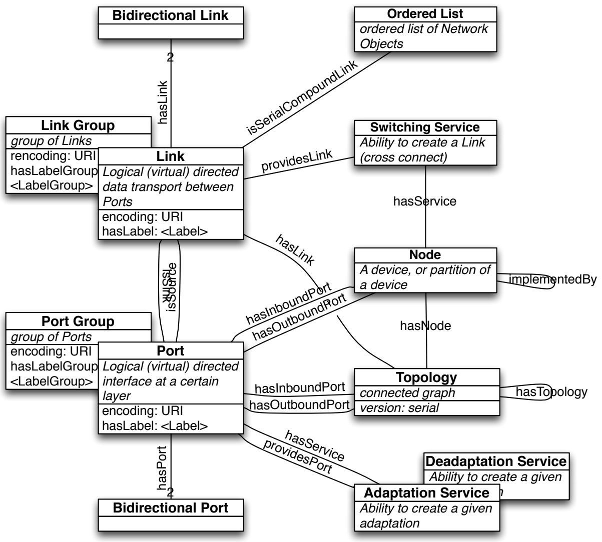


Figure 2.3: Network Markup Language main classes and properties

NML [125] is a generic network description model shown in Fig. 2.3. The basic elements in NML are *Node*, *Port*, *Link* and *Topology*. *Node* is a device in the network, which can be hardware resources – a router, switch or computer machine. A *Node* connects to the network through its *Port*. The *Link* is a connection between two *Ports*. *Ports*, *Links* and *Nodes* make up a network topology. An important difference with previous network models is that NML is a completely unidirectional model. The Port can be logical concept, which does not correspond to one physical interface. One physical interface can have two unidirectional NML port individuals to describe the traffic in different directions.

INDL uses the *nml:Node* concept as the basic entity for describing a resource in a computing infrastructure. INDL can be used as a stand-alone model (i.e. without any network description), or it can be used in combination with NML by importing the NML ontology into the INDL definition. We focus on the latter case in this thesis, all NML concepts will become available to the user of INDL. In the following, we describe the main classes and properties in the INDL ontology.

Fig. 2.4 shows how the internal components of a node are modeled by defining *nml:Node* to consist of a number of *NodeComponent*. The *NodeComponent* is an abstract class which describes essential components of machines which are of interest to the user: *MemoryComponent* shows how much memory is available at a node, *ProcessorComponent* to describe how many cores a node has, their speed, etc. and *StorageComponent* to define the space available for local storage.

Virtualization is modeled using the *VirtualNode* concept, which is modeled as a subclass of *nml:Node* (i.e. a virtual node inherits all properties of a node). A virtual node is also implemented on a node (see Fig. 2.5). The implementing node itself can be either a physical node or another virtual node. This allows us to create layers of virtualization stacked on top of each other.

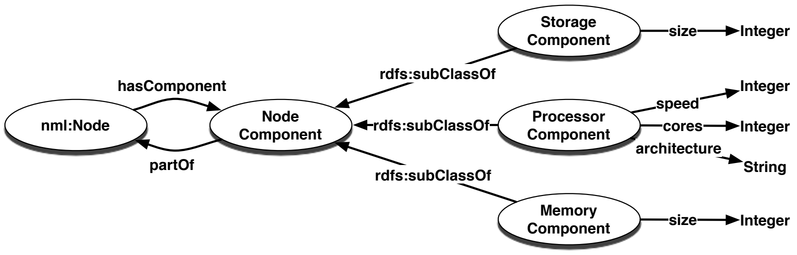


Figure 2.4: INDL: Modeling internal node components.

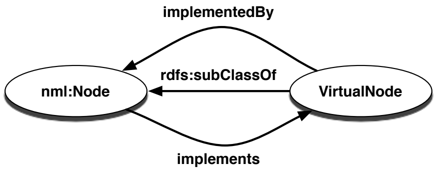


Figure 2.5: INDL: Modeling virtualization of nodes.

The key feature of INDL which makes it reusable and easy to extend is that it decouples virtualization, functionality and connectivity. This allows us to add new functionality (e.g. adding a new type of *NodeComponent*) without impacting how we model its connectivity with other devices or how we model virtualization of the new resource. Furthermore, connectivity and functionality is modeled the same for physical nodes and virtual nodes which allows INDL to describe physical computing infrastructures as well as virtual infrastructures.

The use of Semantic Web technology in INDL and NML ontologies facilitates the creation of models that can be easily stacked and extended by other models. NML and INDL have been used to define the models for three different computing infrastructures: the CineGrid infrastructure [87], the Logical Infrastructure Composition Layer in the GEYSERS EU-FP7 project [68] and the NOVI federation platform [18]. It was also the basis for ExoGENI [20]. In these projects, INDL shows its support for describing distributed infrastructures. Operators in each domain create the description for their infrastructure based on INDL; then they can publish the description in the same information system or on the Web and a resource management system can gather the information needed for global management across domains. An e-Infrastructure can also be described in this way. In essence, INDL is a suitable model for e-Infrastructures.

2.4 Energy Description Language

2.4.1 Energy-aware Information Models

There are a number of information models that have been designed to capture energy-awareness.

Daouadji et al. developed an ontology-based resources description framework for resource allocation purpose with minimal CO2 emissions [57]. The framework is used by GreenStar Network (GSN) [33], the first nation-wide network in the world which is powered only by green energy. In the framework, they proposed a simple ICT energy consumption model (GSNONT) based on the Semantic Web. Each resource has an associated energy type, which is categorized into *Green* and *Brown*. Each type of energy source has a property, which quantizes the CO2 emissions per unit of energy.

The Common Information Model (CIM) [15] is a comprehensive UML model for ICT systems and devices. It was developed by the Distributed Management Task Force (DMTF) [1]. CIM is a very broad and complex model, and the current UML schema of the total model is over 200 pages. Given the complexity of the CIM, it's not easily reused or extended. DMTF has defined a mapping from UML to XML, which is mainly implemented in enterprise-oriented computing infrastructure and operating systems. CIM includes a set of objects related to energy monitoring. The *Power Source* class describes the output power of entities that produce power; *Power Supply* captures the capabilities of input voltage and frequency entities that supply power. *Metric* details the measured value by *Sensor*.

The Green Information Model (GIM) [49] is the outcome of the Mantychore project. GIM uses XML Schema to capture the energy and green considerations of a network, and it includes the concepts of the power delivery, power supply and power monitoring components. Power delivery resources represent any device that directly delivers power to network devices. Each device is associated with a power supply, but a network can use a number of power supplies simultaneously.

The Energy Management (EMAN) [16] working group in Internet Engineering Task Force (IETF) [6] focuses on an energy management framework for IP-based network equipment. EMAN presents an information model for energy monitoring and energy control. Energy monitoring includes measuring power, energy demand and attributes of power. Energy control sets a device's or component's state. The model also addresses the issues of identification and classification of devices in networks.

We compare the features of the available information models in Table 2.1. Namely, we look at

- Domain,
- Object,
- Range,
- Model,
- Renewable energy,
- Relation of sensors and devices,
- Sufficient metrics.

The first three information models only aim at describing information about energy monitoring, and EMAN also targets information about energy control. GSNONT and CIM can describe the states of computing and networking devices as well as their components e.g. CPU and memory, while EMAN and GIM only monitor the networking devices. GSNONT is a quite specialized model, mostly applied for green routing path selection. Only GSNONT is an OWL-based ontology while other models use XML Schema. GIM and EMAN lack the property

Table 2.1: Comparison of existing energy-aware information models in the description of infrastructures.

Name	GSNONT	CIM	GIM	EMAN
Domain	monitoring	monitoring	monitoring	monitoring & control
Object	devices& components	devices& components	devices	devices
Range	green routing	networks &computing systems	networks &computing systems	networks &computing systems
Model	OWL	XML Schema	XML Schema	XML Schema
Renewable energy	$\surd$	$\surd$	$\times$	$\times$
Relation of sensors & devices	$\times$	$\times$	$\times$	$\surd$
Sufficient metrics	$\times$	$\surd$	$\times$	$\times$

that describes sustainability e.g. CO2 emissions of an energy source. Different from performance monitoring, the power is measured on instrumentation devices instead of resources that consume power, and usually one sensor monitors multiple devices simultaneously. This requires a clear definition of the relationship between the instrumentation devices and identification of remote devices for which monitoring information is provided. Currently, only EMAN can support the description of this relationship. Except CIM, other models only support basic energy-related metrics such as power, current or voltage. In fact, the energy metrics of resources can be GHG emission, electricity cost, energy efficiency, etc. The lack of metrics in the models limits the range these models can be applied in. All these models above are not suitable to describe the energy-aware infrastructure across multi-domains for energy management. We will propose our information model in the next section.

2.4.2 Energy Description Language

The goal of EDL is to represent energy monitoring objects and the energy-related states of resources. EDL links them to the *Node* class in INDL. The EDL model is shown in Fig. 2.6; it contains three main parts:

1. The *Green Metric* part defines classes and properties to describe measurement data using different energy metrics.
2. The *Characteristic* part is related to the non-measurable states of computing or networking resources, e.g. energy source and energy efficient capabilities, which support energy-aware resource discovery.
3. The *Monitor Component* part describes the way of obtaining measurement data of resources from sensors and the way of organizing measurement data in logs.

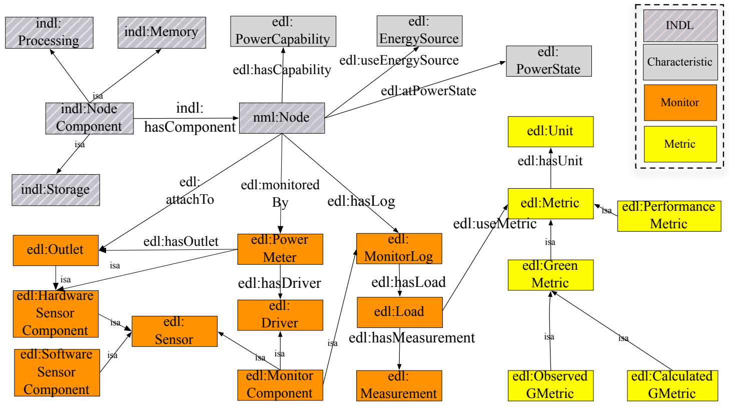


Figure 2.6: Energy Description Language imports INDL – INDL and three main parts in EDL. They are the *Green Metric* classes and their properties, the energy *Characteristic* classes and properties and the *Monitor Component* classes and their properties

2.4.2.1 Green Metric

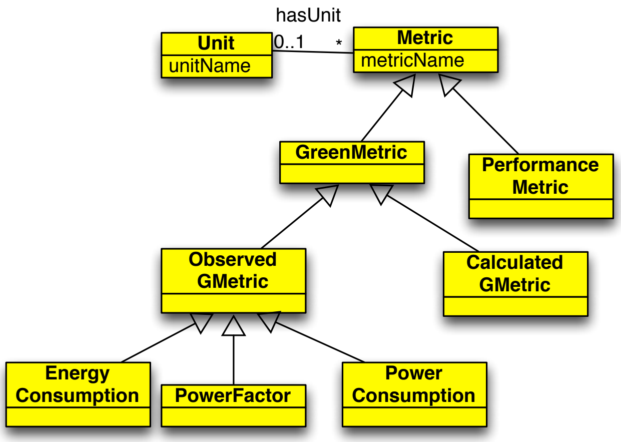


Figure 2.7: UML representation of Green Metric part

From Fig. 2.7, the *Metric* class contains the *Performance Metric* class and the *Green Metric* class. *Performance Metric* can be the general performance metrics such as throughput and utilization. Performance metrics measure the capability of resources. There is already some work on modeling performance metrics [64] [135]. We reuse the common metrics for hardware components from them. In this thesis, we focus on energy metrics.

The *Green Metric* class represents the measurable states of resources in various metrics related to energy and sustainability. Energy management systems leverage the current states using the green metrics to decide how to schedule tasks. The metrics they are interested in could be diverse. Some systems may care about GHG emissions, while others may focus on metrics about the electricity cost of an infrastructure.

The Green Metric class defines two types of energy metric, *Observed GMetric* and *Calculated GMetric*, distinguished by the way measurement data is obtained. Data in the former metrics is directly collected from power meters such as Power Distribution Units (PDUs), while data in the latter is usually obtained by numerical calculations of measurement data using observed green metrics and performance metrics.

In EDL, we predefined some green metrics. Observed green metrics from power meters are usually fixed, so we defined them as classes. These classes can't be added or removed once loaded into information systems. *Power Factor* is the ratio of the real power that is used to do work and the apparent power that is supplied to the circuit. *Energy Consumption* and *Power Consumption* represent the total energy consumption in a period of time and real-time power consumption, respectively.

The calculated metrics are numerous. We defined the calculated metrics as individuals which can be dynamically instantiated using *Calculated GMetric* class by users. *Energy Efficiency* and *Emission Efficiency* are two calculated metrics already defined in EDL. Energy Efficiency is a measure of the rate of computation or transmission that can be processed by a computer for every watt of power consumed. The Green500 ranks supercomputers in the Top500 list using FLoating-point Operations Per Second (FLOPS) per Watt [30]. Also, Energy Efficiency measures the number of operations or transmissions for every joule of energy consumed. Comparably, Emission Efficiency measures the number of operations or the bytes of data transmission for every unit of GHG emissions. *Calculated GMetric* also includes the metrics of the overall infrastructure like PUE. We also define the total volume of GHG emissions and total electricity cost.

Although the efficiency metrics seem more useful, absolute metrics, e.g. Power Consumption are essential. Energy Efficiency can be improved by enhancing the performance even if resources continue to consume large amounts of absolute power. Resources in the idle state can not be adequately characterised by just efficiency but can be evaluated by measuring Energy Consumption and Power Consumption.

Each metric individual is associated with a *Unit* according to its physical quantity. In many cases a numerical value alone cannot be understood without its unit type. We list the predefined green metrics and their units in Table 2.2.

Table 2.2: The predefined green metrics and their units in the EDL ontology

Group	Metric	Unit
Observed Metrics	Power Factor	-
	Energy Consumption	Joule (J)
	Power Consumption	Watt (W)
Calculated Metrics	Energy Efficiency	Bytes per Joule
	Emission Efficiency	Bytes per gram

2.4.2.2 Energy-aware Characteristics

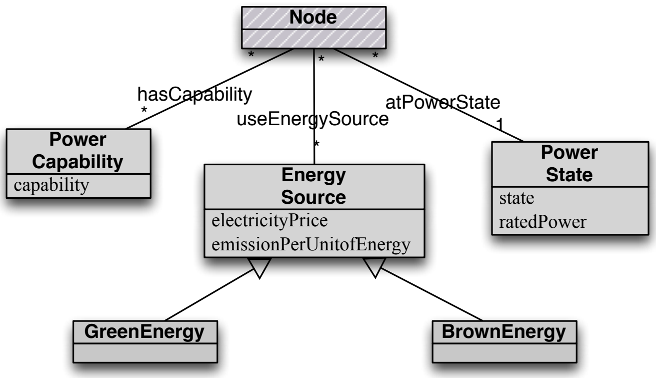


Figure 2.8: UML representation of energy-aware Characteristic part

The UML representation of energy-aware characteristics is shown in Fig. 2.8. The *Energy Source* class defines the type of energy source used by the resources in infrastructures, *e.g.* wind, solar or thermal. Each energy source has a corresponding electricity price and GHG emissions rate; these can be used to calculate the total GHG emissions of the resources in a period of time. With the description of energy sources, EDL has an awareness of environmental sustainability of resources.

The running state of a resource is determined by the *Power State* class. A management system should know whether the resource is in *Off*, *Sleep* or *Active* state. The *ratedPower* property describes the power of a device under nominal (idle) operating conditions. The *Power Capability* class indicates what low power capability the resource has. Resources that are made up of embedded processors like Intel Atoms, Solid State Disk (SSD) storage and Energy Efficient Ethernet supporting IEEE 802.3az [11] have low power or energy efficient capability.

Based on the description of energy characteristics in EDL, applications can discover resources from the knowledge base when given specific requirements for energy sources, energy states or power capabilities. For example, EDL allows applications to discover resources with green energy or with low power processors or GPUs.

2.4.2.3 Monitor Component

The most significant difference between energy monitoring and performance monitoring is that energy monitoring needs extra instrumentation devices to determine power or energy. The power is measured on these instrumentation devices instead of directly on resources. For example, power value of a resource is retrieved from a PDU at the outlet. Therefore, the relationship between resources and instrumentation devices must be understood.

As shown in Fig. 2.9, the relationship between resources and instrumentation devices is described by *Hardware Sensor Component* under the *Sensor* class.

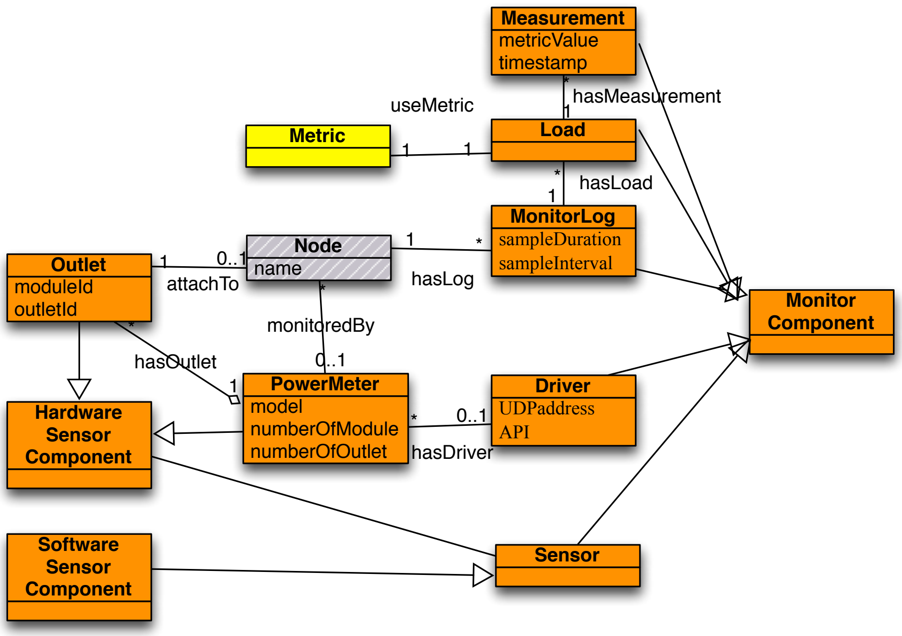


Figure 2.9: UML representation of Monitor Component part

Sensors include hardware sensors and hardware sensors. The *Software Sensor Component* is software systems or tools which monitor the performance attributes that are not available from hardware sensors.

The *Power Meter* class represents the instrumentation device, usually a PDU, for energy monitoring. Each node can be monitored by a PDU, which usually has multiple modules. Each module includes multiple outlets that attach to different resources. Each specified *Outlet* is only responsible for providing the measurement data of the resource attached. The property *attachTo* describes a one-to-one mapping between a resource and a outlet of the power meter. PDUs are differentiated by model or type, which feature an access address and APIs to collect measurement data. The *Driver* of PDUs describes this information.

Besides the relationship of resources and power meters, the *Monitor Component* describes the organization of measurement data. The *MonitorLog* is a collection of *Load* instances in different metrics with additional properties about sampling time interval as well as the start time and end time of sampling. The load has Measurement data for the same metric. Each Measurement individual in a load represents one measurement that has an *metricValue* modeled by a *datatype* properties with *xsd:double* type value. The measurement has a data property of *timestamp*. The value of timestamp represents the time at which the measurement was taken.

2.4.3 Features of Energy Description Language

In Table 2.3, we summarize the features of EDL. This should be compared with Table 2.1 where we reported other previous models. EDL can describe fine-grained resources and virtualized resources given that it leverages INDL to model internal components of nodes and virtualized nodes. It provides the description for networks and computing systems. EDL is a semantic model that improves data interoperability and stays extensible. EDL can describe renewable energy, and it can represent the relations between power meters and devices. The measurement data can be described by EDL using any metrics.

Table 2.3: Features of Energy Description Language in the description of infrastructures.

Name	EDL
Domain	monitoring
Object	(virtualized) devices& components
Range	networks & computing systems
Model	OWL
Renewable energy	√
Relation of sensors & devices	√
Sufficient metrics	√

EDL decouples the energy-related states from the configuration and structure of resources. This allows us to add new energy-related states, for example adding the location of an energy source without influencing how we update the connectivity and the composition of resources in EDL. There are an additional features of EDL that make it a very strong model for energy management.

EDL supports a wide range of energy management scenarios:

1) EDL supports the power estimation. It defines an energy measurement log, which allows the representation of measurement data in different metrics at different sample time intervals. This measurement data is necessary for various energy management scenarios. One of most important is the analysis of measurement data statistics which can support construction of power estimation models. These models are used to predict the power consumption of scheduling decisions.

2) EDL can describe the agreements between operators and users, as it defines energy cost and capacity of resource components. Energy budget accounts for a significant part of cost of operating infrastructures. On Clouds or Grids, the price that operators can charge depends on the performance they advertise; but the energy consumption of these resources is diverse. Cloud or Grid users can be encouraged to wait to utilize a combination of energy efficient resources with low performance requirements, and in the meantime the operators refund part of the profits from the cost of saving energy. But users are worried about whether the amount of refund is real and is worthwhile. They need to reach agreements with the operators.

The operators use the classes and properties in EDL to design an automatic mechanism for disseminating claims about resource capacity and energy cost in the agreements. The users have an explicit EDL model and are therefore capable of understanding these agreements. In this way, online agreements can be easily reached at a low-cost.

3) EDL is able to support green resource discovery across different domains, finding green networks or computing resources. Although some infrastructures or parts of resources have been using green energy source, there is no method for differentiating them from other infrastructures or resources, limiting the effect of power management. Energy-related states such as the type of energy source of resources and the GHG emission rate generated are abstracted in EDL. It enables operators to search this information across domains in order to discover resources in terms of their energy and sustainability state.

2.5 Conclusion

In this chapter, we discussed the motivations for using a semantic approach to describe an e-Infrastructure with energy-awareness. RDF improves data interoperability in a distributed environment and OWL ontologies are extensible so that we can easily build new models based on our previous work: the Energy Description Language imports the Infrastructure and Network Description Language.

We first presented INDL, its classes and properties. And we showed how this model captures the concept of virtualization and describe the capabilities of hardware components in computing and network infrastructures.

After that, we gave an overview of the features of existing energy-aware information models for describing infrastructures. None of these models are fully suitable for fine-grained description of infrastructures across multiple domains. Therefore we developed EDL, a semantic model which defines metrics that describe energy footprint in relation to workloads, non-measurable state of resources and objects used for energy monitoring. EDL can model internal components of a node and virtualized nodes. EDL can describe renewable energy, and it can represent the relations between power meters and devices. The measurement data can be described by EDL using any metrics. The concepts in EDL can support a wide range of power management scenarios. In the next chapter, we will show an energy-aware information system – the Energy Knowledge Base system [137]. EKB has adopted the EDL ontology to properly organize and manipulate energy-related knowledge.

Chapter 3

The Energy Knowledge Base System

—A semantic information system for energy monitoring in e-Infrastructures

This chapter is based on:

- **H. Zhu**, K. van der Veldt, P. Grosso, X. Liao and C. de Laat (2013). EKB: a semantic information system for energy-aware monitoring in distributed infrastructures. In Proceedings: 2013 IEEE Third International Conference on Cloud and Green Computing: CGC 2013, pages 60-67, Karlsruhe, Germany.
- **H. Zhu**, K. van der Veldt, Z. Zhao, P. Grosso, D. Pavlov, J. Soeurt, X. Liao, and C. de Laat (2015). A semantic enhanced power budget calculator for distributed computing using ieee 802.3az, Cluster Computing, 18(1), 61-77, 2015.

3.1 Introduction

Distributed Grid or Cloud e-Infrastructures usually integrate an enormous amount of computational, storage and networking resources, data and services from different administrative domains worldwide [84]. For example, EGI [2], the successor to the EGEE projects [10], is a global collaboration of more than 350 resources centers in 56 countries. It provides more than 320,000 logical CPUs and 180 PB of disk space with over 1.7 M compute jobs processed every day in them.

Resource management for such large-scale e-Infrastructures depends on the availability and accuracy of information about individual domains, such as resource type and resource state, and about the interconnections between infrastructures, such as networking topology. There are some information systems that can support this. The Monitoring and Discovery System (MDS) [116] developed for Grid e-Infrastructures like EGEE and EGI collects and aggregates information about resources and services. In terms of network infrastructures, PerfSONAR [75] and SoNoMA [81] are information systems that publish performance information about networking elements using different measurement tools.

However, despite the essential role of energy management to cut the energy and running cost for large-scale infrastructures, existing information systems hardly support it. First, complete information on the energy profile of different applications is missing. Second, information systems that also possess a resource discovery mechanism usually cannot cope with the requirements of applications on energy-awareness. For instance, they cannot meet the requirements for discovering resources powered by renewable energy sources.

Building an energy-aware information system for e-Infrastructures required us to overcome some challenges: 1) the system needs to span across multiple domains of an e-Infrastructure; 2) and it needs to capture and combine information from different sources considering that performance and power information is usually measured on different devices.

In this chapter, we present the architecture of Energy Knowledge Base (EKB); we show how EKB addresses these challenges and embraces more features suitable for e-Infrastructures. Sec. 3.2 compares existing information systems with EKB. In Sec. 3.3 we present the general EKB architecture. Sec. 3.4 shows the applications of EKB for energy management. Sec. 3.5 provides a performance evaluation of two system implementations of the EKB architecture. Finally, Sec. 3.6 concludes this chapter.

3.2 Information Systems

3.2.1 XML-based Information Systems

MDS [116], Berkeley DB Information Index (BDII) [109], Advanced Resource Connector (ARC) [66] and RGMA [55] are developed for Grids and are deployed in systems such as NorduGrid, Europe Data Grid, Crossgrid, and Open Science Grid; they are also used in Clouds now.

Globus MDS4 publishes information about available resources on the Grid and their states. It adopts the Web Service Resource Framework (WSRF) to define an XML-based service information model and standard interfaces for access to service data for clients. The system provides multiple-layer index services to subscribe and cache resource properties (sets of information about a resource) from specified information sources like Ganglia [100]. The trigger service doesn't only subscribe resource information but also performs actions when the collected data meet certain conditions.

BDII and ARC both rely on the Lightweight Directory Access Protocol (LDAP) server and are derived from the Globus MDS2 framework. They use the same GLUE information model and access APIs. They provide resource information by doing an ldapsearch on LDAP URLs.

PerfSONAR [75] and SONoMA [81] are used for monitoring and measuring wide-area network infrastructures such as GLIF [4] or GÉANT. The Measurement Points (MPs) in PerfSONAR wrap network tools like Ping or Iperf for measuring network metrics. PerfSONAR consists of a couple of web services, e.g. Lookup Service and Measurement Archive (MA) Service, which can respond to queries about performance measurement and discover service registered in a federated environment. It provides visualization tools and service interfaces to perform tests. SONoMA is a network measurement architecture, which has similar MP and MA services with PerfSONAR. But SONoMA uses a different information model, and lacks the capability of monitoring a network across multiple domains.

The information systems above are not designed for energy monitoring, and they lack the functionality of describing and collecting energy information. The Cisco EnergyWise [54] framework aims to obtain power measurement data. It allows networks to advertise the power consumption of attached devices using a network-wide approach and a query mechanism via the Simple Network Management Protocol (SNMP). But this way of relaying information between neighbouring devices is not suitable for large scale infrastructures, where the number of monitored resources across different domains is huge and the response time would become unacceptable. Moreover, the metrics for qualifying energy consumption in EnergyWise are too simple and cannot be extended.

All these information systems described here are built on XML-based information models. This limits possibilities of sharing data and extending models.

3.2.2 Semantic Information Systems

There is ongoing research on semantic systems for Grids and Clouds, which provide advantages of data interoperability.

The Semantic Grid seeks to incorporate the Semantic Web approach into the ongoing Grid. Using RDF and ontologies can offer high-level support for managing Grid resources through exposing metadata in an interoperable form. The Semantic Open Grid Service Architecture (S-OGSA), extends OGSA by defining a lightweight mechanism that allows for the explicit use of semantics alongside the associated knowledge services to support a spectrum of service capabilities [56].

The Semantic Monitoring and Discovery System (S-MDS) [115] is built on top of the Globus MDS4 toolkit. S-MDS dynamically creates an information model from MDS resource properties files and develops algorithms for automatic ontology instantiation. It includes an index service, which is similar to the one in MDS to organize and publish semantic data. The system provides some GUI tools for creating, enriching and registering semantic metadata, and constructing SPARQL queries [111]. However, information model creation and ontology instantiation algorithm depend on resource properties files, and this therefore follows that MDS is the only one available information source for S-MDS. It cannot adapt to other information sources and cannot be extended to include power monitoring.

ActOn [130] is an ontology-based information integration system for Grids. Metadata Cache provides fast access to information that has been already integrated and materialized. It focuses on information integration, and allows the selection of Grid information sources such as BDII and MDS to generate and maintain up-to-date metadata. It neglects research on information models for metadata description. And it only works locally without the capability to aggregate information across different domains.

Table 3.1 summarizes the information systems above, and compares the differences between them and EKB. Namely, we look at

1. Semantic,
2. Energy-aware,
3. Support multiple domains,
4. Scalable,
5. The number of information source,

Table 3.1: Features of Energy Knowledge Base and common information systems.

System	MDS	Perf SONAR	Energy Wise	S-MDS	ActON	EKB
Semantic	×	×	×	√	√	√
Energy-aware	×	×	√	×	×	√
Multi-domain	√	√	×	√	×	√
Scalable	√	×	×	√	√	√
Info source	multiple	multiple	single	single	multiple	multiple
Range	Grids /Clouds	networks	networks	Grids /Clouds	Grids /Clouds	networks &Grids /Clouds
Info model	XML: GLUE	XML: NMWG	–	OWL: Dynamic	OWL: DO	OWL: EDL

6. Range,
7. Information model.

MDS, S-MDS and ActON are systems for monitoring Grids and Clouds. MDS is not semantic system, ActON can't work in an infrastructure with multiple domains and MDS is the only available information source for S-MDS. PerfSONAR and EnergyWise are monitoring system for network infrastructures. They don't leverage Semantic Web technologies. PerfSONAR can monitor networks across different domains. PerfSONAR can collect information from different type of networking tools while EnergyWise can only collect the state information of devices via SNMP. Apart from EnergyWise, all the other systems are not energy-aware. But inserting capabilities for energy monitoring in them is a prohibitive task because all operations in these information systems are closely related to models that are not energy-aware either. We design a new information system based on the EDL ontology. We introduced the Energy Knowledge Base because it is a semantic and energy-aware information system. It can work across multiple domains of Grids/Clouds and networks. It supports interactions with different information sources. Our evaluation will show its performance is scalable with the increase of users.

3.3 The Architecture of Energy Knowledge Base

The EKB system is a semantic information system that can be used to generate and maintain energy-related metadata and measurement data for an e-Infrastructure. Measurement data here mainly means performance or energy usage data. Metadata means all the other information about infrastructures such as configuration, structure and topology. The development of EKB was driven by a series of desired features:

- *Interoperable*: Information from different administrative domains can be shared and reused.

- *Distributed*: EKB should have a distributed structure that fits an e-Infrastructure spanning multiple administrative domains; a centralized EKB is not efficient and robust. EKB maintains information in each local domain and provides the information to peers in other domains to form knowledge of the overall e-Infrastructure.
- *Support for multiple information sources*: Information sources such as Ganglia and MDS for monitoring infrastructures provide actual configuration, structure and performance information, and power meters provide energy information. EKB should be able to combine different information sources to provide the complete information.
- *Scalable*: EKB should quickly answer information requests from local or remote domains, and keep the response time low even when the number of the users increases.

EKB has been the adoption of Semantic Web technologies that facilitate the energy information interoperability. The EKB system leverages the Energy Description Language to model the configuration and structure of computing and network resources as well as their energy-related states. Knowledge is represented by RDF triples in EKB.

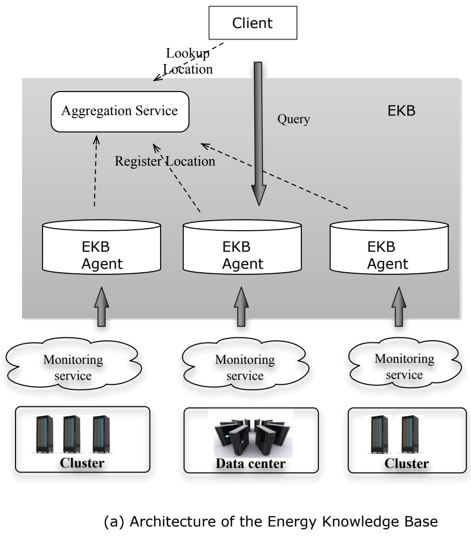
Besides the benefits mentioned in Sec. 2.2.2, we see an additional advantage in grounding EKB in the Semantic Web. Semantic Web technologies facilitate energy information retrieval. In e-Infrastructures, sophisticated power management systems consider various resource capabilities and states before scheduling. But these systems either do not support some kind of energy-aware resource discovery or only support keyword based discovery of resources. A semantic inference engine can infer potential relationship. For example, a semantic inference engine with a *subsume* capability [117] can infer relationship between energy sources to return possible candidates while keyword based resource discovery often miss relatively capable resources.

Fig. 3.1 (a) shows the EKB architecture. EKB contains multiple agents that are connected by the Aggregation Service. Fig. 3.1 (b) shows an EKB agent is composed of knowledge components, which represent the knowledge from the application domain and from information sources; and software components, which collect, store and provide the knowledge.

3.3.1 Aggregation Service

To have a distributed EKB, EKB agents should be able to cooperate with each other to publish energy information and discover resources. We achieve this capability through the Aggregation Service. We build an EKB agent for each domain. These agents are disconnected in the beginning. Then, the EKB agents register with the Aggregation Service. The Aggregation Service maintains a global list of registered agents in an e-Infrastructure. Clients can discover EKB agents by querying the list, which is similar to the Lookup Service (LS) component of the PerfSONAR [75].

A global list is an easy way to connect all the distributed agents. However, this design is not suitable for large-scale infrastructures as the discovery of an EKB agent from a long list with thousands of agents might becomes bottleneck. We



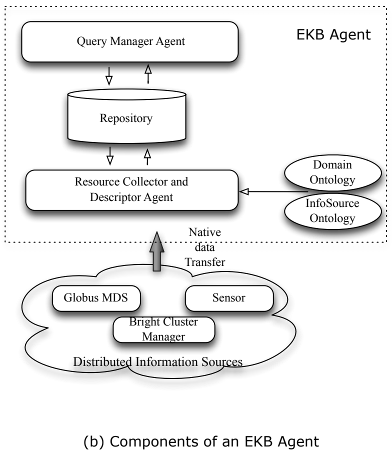


Figure 3.1: (a) Overview of the Energy Knowledge Base architecture (b) knowledge components and software components of an EKB Agent

didn't have the bottleneck as EKB is mainly designed for DAS-4 cluster system that is not a production infrastructure. In a future implementation for large-scale infrastructures, a Peer to Peer (P2P) mechanism for agent discovery would be a better choice for EKB. We have already deployed two EKB agents on the clusters located at the University of Amsterdam and VU University Amsterdam respectively.

3.3.2 The EKB Knowledge Components

One of the main ingredients in our design and implementation is the knowledge components. The EKB knowledge component contains a domain ontology and an ontology about information sources.

The domain ontology describes the information model in the form of domain concepts and properties for which instances will be generated. EDL is used as the domain ontology to describe complicated information about infrastructures and their energy-related states. As EDL imports INDL, EKB can not only describe computing infrastructures such as Grids/Clouds but also describe network infrastructures. All measurement data and metadata about infrastructures is instantiated by the EDL ontology.

We design a simple information source ontology to describe common concepts among different information sources. The ontology only contains data about the location and identity of information sources like IP addresses and certificate file URIs. For each information source, a description file is created based on this ontology. EKB can interact with different information sources according to the location and identity information in these description files.

3.3.3 The EKB Software Components

There are three software components in an EKB agent: the Query Manager Agent (QMA), the Repository, and the Resource Collector and Description Agent (RCDA).

3.3.3.1 Query Manager Agent

The Query Management Agent receives the web queries from clients and translate the queries into proper SPARQL queries.

SPARQL is usually used for information retrieval from a knowledge base. But for users who have no experience of SPARQL, writing a SPARQL query is not trivial. To increase the convenience and robustness of the information system, we do not allow users to directly send SPARQL queries, instead we define REST-based web APIs, and web requests are transparently translated into SPARQL queries in the QMA.

Table 3.2: Part of the REST APIs in the Energy Knowledge Base

Type of information	Information	Example	APIs (/topology/...)
Measurement data	Energy	Power consumption, energy consumption, etc.	/pdu/data/<pdu>/<outletnum> ?metrics=&start=&end=
	Performance	CPU utilization, Mmmory utilization, etc.	/nodes/<node>/data /<component> ?metrics=&start=&end=
Metadata of infrastructures	Nodes	Nodes of in the infrastructure	/nodes
	Metrics	Available metrics for measurement data	/metrics
	Node components	CPU frequency, memory and disk size, etc.	/nodes/<node>/attribute /component
	Power meters	Connection between power meters and nodes	/pdu/mapping
	Networks	Ports of nodes	/nodes/<node>/attribute/port
	Networks	Link between ports	/nodes/<node>/port/<port>
	Energy source	Type of energy source	/nodes/<node>/attribute /energysource

We list part of the REST APIs in EKB in Table 3.2. The applications can query energy and performance measurement data. EKB also supports querying the metadata of the infrastructure, which includes the available metrics, the capabilities of networks and computing nodes, the mapping information of power meters to nodes, the energy sources, etc.

3.3.3.2 Repository

The Repository is a persistent database system that stores the data obtained from the underlying RCDA and responds to requests from the QMA. In our initial system design, we only used the Sesame [47] triple store to maintain all the measurement data and metadata of a cluster.

However, we faced performance problems with retrieving measurement data, and we will show this in Sec. 3.5. In an alternative implementation of the Repository, it stores metadata in a triple store, but stores measurement data as time series in a high-performance time series database (TSDB) called KairosDB [76].

Most of the data in our system by far is measurement data with time-series, so this greatly improves the performance of the overall system. In addition, it also simplifies time-series related tasks, such as downsampling. Each sample of measurement data is not as highly connected with each other as in a RDF graph, and samples can be described in a simple relationship. We don't represent measurement data in RDF triples, but this hardly sacrifices data interoperability . In this implementation, QMA translates received web queries into TSDB queries instead of SPARQL queries when measurement data is requested.

3.3.3.3 Resource Collector and Description Agent

The RCDA collects and instantiates native data in order to have organized knowledge with high interoperability. Before that, EKB obtains location and identity information from the description files of information sources. With this information, EKB can access multiple information sources to collect the complete information.

EKB obtains metadata of infrastructures from the underlying information system, and it collects energy measurement information from power meters. In the current EKB implementation, information sources are limited and fixed, only including Ganglia [100] and PDUs. We can figure out what APIs they have and what kind of information they provide, so EKB can automatically instantiate native data.

The RCDA saves instantiated data into the Repository. The instantiated data is either semantic description or measurement data. The following is an example of the EDL description of a computing node and its energy-related states. This node has a 6-core processor with 2.0 GHz and 64G memory (line 34-39, 54-56). It consumes wind energy and its energy is monitored by "rpdu-vu1" PDU (line 73-79) and "outlet1" (line 63-67). The access address of rpdu-vu1 is "10.148.0.252:161" (line 84-89).

```
 1  <!-- http://www.science.uva.nl/research/sne/edl#HadoopNode01 -->
 2
 3      <NamedIndividual rdf:about="&edl;HadoopNode01">
 4          <rdf:type rdf:resource="http://schemas.ogf.org/nml/2013/05/base#Node"/>
 5          <edl:useEnergySource rdf:resource="&edl;Wind"/>
 6          <indl:hasComponent rdf:resource="&edl;Xeon1"/>
 7          <indl:hasComponent rdf:resource="&edl;memory64"/>
 8          <edl:attachTo rdf:resource="&edl;outlet1"/>
 9          <edl:monitoredBy rdf:resource="&edl;rpdu-vu1"/>
10      </NamedIndividual>
11
12
13
14      <!-- http://www.science.uva.nl/research/sne/edl#HadoopNode1 -->
15
16      <NamedIndividual rdf:about="&edl;HadoopNode1">
17          <edl:nodeName rdf:datatype="&xsd;string">node091</edl:nodeName>
18      </NamedIndividual>
19
20
21
22      <!-- http://www.science.uva.nl/research/sne/edl#Wind -->
23
24      <NamedIndividual rdf:about="&edl;Wind">
25          <rdf:type rdf:resource="&edl;GreenEnergy"/>
26          <edl:emissionPerUnitofEnergy rdf:datatype="&xsd;float">11.0
27            </edl:emissionPerUnitofEnergy>
28      </NamedIndividual>
29
30
31
32      <!-- http://www.science.uva.nl/research/sne/edl#Xeon1 -->
33
34      <NamedIndividual rdf:about="&edl;Xeon1">
```

```
        <rdf:type rdf:resource="&indl;ProcessingComponent"/>
        <indl:cpuspeed rdf:datatype="&xsd;float">2.0</indl:cpuspeed>
        <indl:cores rdf:datatype="&xsd;int">6</indl:cores>
        <indl:cpuarch rdf:datatype="&xsd;string">Sandy Bridge</indl:cpuarch>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#emissionRateUnit1 -->

    <NamedIndividual rdf:about="&edl;emissionRateUnit1">
        <rdf:type rdf:resource="&edl;Unit"/>
        <edl:unitName rdf:datatype="&xsd;string">g/kWh</edl:unitName>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#memory64 -->

    <NamedIndividual rdf:about="&edl;memory64">
        <rdf:type rdf:resource="&indl;MemoryComponent"/>
        <indl:size rdf:datatype="&xsd;long">64</indl:size>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#outlet1 -->

    <NamedIndividual rdf:about="&edl;outlet1">
        <rdf:type rdf:resource="&edl;Outlet"/>
        <edl:outletId rdf:datatype="&xsd;int">1</edl:outletId>
        <edl:moduleId rdf:datatype="&xsd;int">1</edl:moduleId>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#rpdu-vu1 -->

    <NamedIndividual rdf:about="&edl;rpdu-vu1">
        <rdf:type rdf:resource="&edl;PowerMeter"/>
        <edl:numberOfModule rdf:datatype="&xsd;int">3</edl:numberOfModule>
        <edl:numberOfOutlet rdf:datatype="&xsd;int">8</edl:numberOfOutlet>
        <edl:model rdf:datatype="&xsd;string">Rackitivity</edl:model>
        <edl:hasOutlet rdf:resource="&edl;outlet1"/>
    </NamedIndividual>


        <!-- http://www.science.uva.nl/research/sne/edl#rpduDriver -->

    <NamedIndividual rdf:about="&edl;rpduDriver">
        <rdf:type rdf:resource="&edl;Driver"/>
        <edl:UDPaddress rdf:datatype="&xsd;string" >10.148.0.252:161
                        </edl:UDPaddress>
        <edl:hasDriver rdf:resource="&edl;rpduDriver"/>
    </NamedIndividual>


</rdf:RDF>
```

The format of measurement data in the TSDB is quite simple, and each record of measurement is in key-value pairs of *"name": metric, "timestamp": timestamp, "value": value, and "tags": tags.* The tags are used to describe where data is measured, e.g. on nodes or PDUs.

We update semantic description and measurement data in different frequencies. Semantic description only needs to be updated occasionally, as the described infrastructure does not change that often, but measurement data is continuously collected at regular intervals. This up-to-date mechanism is efficient compared to always updating all the information frequently.

3.4 The Applications of Energy Knowledge Base

The reason that EKB maintains the measurement data and metadata is they are required by energy management in two common scenarios from the view of resources: measurement data required by energy-aware resource allocation; and metadata by energy-aware resource discovery.

Energy-aware resource allocation maps workloads onto resources of an infrastructure in an energy efficient manner. As workload information is given, operators need to estimate energy consumption of resources as a result of the resource allocation to be carried out. When this energy information is known, operators can find an effective solution, which uses the least energy consumption but satisfies any SLAs. The measurement data in EKB includes power and performance data. A power estimation model can be studied from the historical measurement data.

Grid/Cloud operators often deal with requests to discover resources with different software and hardware configuration, for example, discovering a low-power server with energy-efficient embedded processors, or finding six network nodes powered by renewable energy sources. Energy-aware resource discovery doesn't need effective resource allocation, and it focuses on meeting those resource requirements of users. Even so, discovering suitable resources that match the requirements is really a difficult task. EKB includes semantic description of energy-related configuration of resources. EKB can discover resources using SPARQL queries when the configuration or energy-related states of resources are given.

In this section, we will discuss two energy management usecases to show how the EKB system supports energy management in the scenario of resource allocation and resource discovery.

3.4.1 Semantic-Enhanced Power Budget Calculator for Infrastructures Using IEEE 802.3az

One of the technologies that optimize the efficiency of networking equipment is putting active switch ports to sleep when no traffic is expected [11]. This technology has become the core technology of the Energy Efficient Ethernet (EEE), and it has recently been standardized as the IEEE 802.3az protocol in 2010. Instead of shutting down unused ports, 802.3az provides flexible mechanisms for customizing device energy consumption based on network traffic. It offers applications the opportunity to consider its energy consumption by tuning its communication loads while scheduling computing tasks; however, it also requires profound understanding of 802.3az energy behaviour given different network patterns. We did an investigation of the energy profile of 802.3az switches under different scenarios and using different traffic patterns [139].

For example, the result of an experiment when using 1Gbit link and *TCP* traffic for a Huawei switch [5] is given in Fig. 3.2. The switch shows a linear increase in power consumption to maximum power consumption at around 450Mbps.

Using the investigation results, we devised a *Power Budget Calculator (PBC)* [1] service for estimating the power usage in clusters equipped with 802.3az-enabled

[1]The current prototype *Power Budget Calculator (PBC)* implementation is available at https://github.com/zupper/cluster-efficiency.

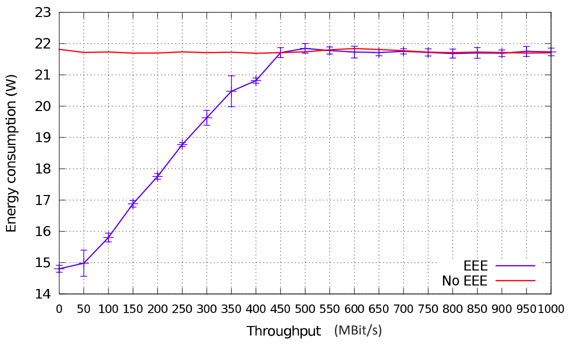


Figure 3.2: Power consumption against TCP flow throughput for Huawei S1728GWR-4P [5].

switches. We estimate the total power usage of a switch as a function of the number of computing nodes and the time needed for a task's completion (*Task-based Estimate*). Using the task-based estimation, the PBC can operate in the context of parallel computing and the output is a recommendation on a power optimal parallelization of a certain task.

The related parameters of the PBC are: the number of available nodes, the switch's power profile, usable ports per switch, total time to complete the task on one node, communication overhead time, network transmission speeds, and average power consumption per node. By analyzing the input parameters and performing time distribution estimation, PBC estimates the total energy usage of the switch for a particular task as a function of the number of used ports on a switch.

Fig. 3.3 shows example output of the task-based estimation tool using the parameters shown in Table 3.3. We can see from the result that using 8 compute nodes would be best for this concrete task if considering the networking infrastructure alone. Typically most power is used for computation, not the network. However for cases where the amount of computation is small compared to the data to be transferred, for example in SKA, only considering the networking infrastructure is possible.

In the input information that PBC consumes, EKB can offer the configuration information and the power profile. The configuration information that includes the available nodes and usable ports of switches can be known through the APIs for metadata in Table 3.2.

EKB supports queries of the power profile across multiple domains as follows.

1. The power profile of a 802.3az compliant device was created and stored in EKB during the period of experiments.
2. The users obtain the address of an EKB agent in the target domain through the Aggregation Service.
3. They check the available metadata information for obtaining measurement data by sending web queries according to Table 3.2. They request the mapping information of power meters to nodes, followed by a query for the metrics available for this node.

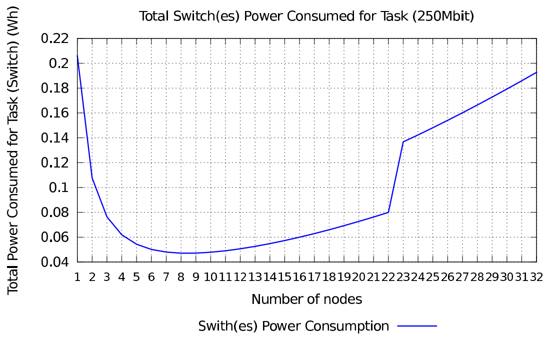


Figure 3.3: Total switch(es) power consumed for task (250Mbit)

Table 3.3: Parameters used for the task-based estimation tool

Parameter	Description	Values
Nodes	the number of available nodes that *could* be used for this task	2
Switch	the switch's profile to use for energy consumption values	Huawei
Ports/switch	usable ports per switch	23
Total time	the total time in minutes needed to complete the task on one node	1
Communication cost	communication cost to exchange messages between nodes	0.05
Speeds	the list of network transmission speeds to use	250
Node power	the average power a node uses while computing	65

4. The PBC can obtain the power profile by specifying the start time and end time of the experiments in the query according to the APIs for measurement data in Table 3.2.

Fig. 3.4 shows the power and performance data obtained from EKB for the *Node079* in a DAS-4 cluster. In this case we select the following metrics among the ones available: power consumption, CPU and memory utilization, IO time of a disk and bytes transferred per second in a network interface card.

EKB enhances Power Budget Calculator from two aspects. 1) EKB improves the efficiency of PBC. PBC doesn't need to maintain power profiles. The EKB system automatically collects data in the backend during experiments, and complements PBC with necessary information at runtime. 2) EKB improves the data quality of PBC because data in EKB can be shared and reused across multiple domains.

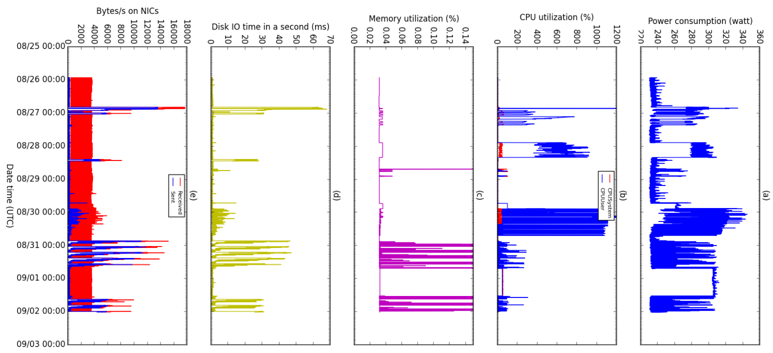


Figure 3.4: An example of measurement data obtained from EKB for a node in DAS-4: (a) power consumption, (b) CPU utilization, (c) memory utilization, (d) disk IO time and (e) network speed

3.4.2 Green Resource Discovery

Table 3.4: A SPARQL query example: find the type of energy source a node is using

```
Query:
SELECT ?nodename ?energysource
WHERE {
?node rdf:type nml:Node.
?node nml:name ?nodename.
FILTER regex(?nodename, 'node079')
?node edl:useEnergySource ?energysource}
```

EKB can be used to discover green resources such as a server with energy-efficient embedded processors, or to find a network path from A to B where (part of) the intermediate nodes are powered by renewable energy sources. EKB supports a request to find out which type of energy source a network node is consuming. The triple store in the backend of EKB checks the type of energy source that the node is using by a SPARQL query (see Table 3.4). Conversely, EKB also supports a request of finding out which node in a network is using a renewable energy source. To be more precise, the emission rate – the amount of GHG emissions per unit of this type of energy – can be known, as it is defined as a data property of the energy source.

3.5 Performance Evaluation

We were interested in the performance of our system. We compare two EKB implementations: EKB-TS uses a triple store to save all data; EKB-TSDB uses a triple store and an extra TSDB where TSDB only saves measurement data.

3.5.1 Experiment Design

We kept the two EKB implementation running on the same physical machine for one week to collect a dataset about 12 nodes. We focus on two performance metrics:

1. Processing time for searching data, *i.e.* the average time (in seconds) to deal with a request inside the database (searching data).
2. Response time for requesting and transferring data, *i.e* the average amount of time (in seconds) for a client to receive a response to a request.

Processing time represents the performance of the database. Response time represents the performance of the server in the QMA and database in the Repository; We designed a variety of queries for measurement data and metadata of infrastructures:

- Query1 requests the power consumption data of one node over the last one hour.
- Query2 requests the power consumption data of one node over the last 24 hours.
- Query3 requests a list of all the nodes in the cluster, which is a metadata query.

At the same time, we simulated hundreds of users to evaluate the scalability of EKB. Each user creates a client to send successive queries with a one second wait between each query.

3.5.2 Results

Fig. 3.5 shows the processing time against different number of users for Query1 and Query2. We simulate 10, 50, and 100 users respectively. EKB-TSDB is at least ten times faster than EKB-TS when requesting measurement data. Both of them spend longer time on Query2 than Query1. The time spent on answering the queries does not seem to depend on the number of clients. The TSDB database has greater capability when processing these two types of measurement queries; this is due to its optimized mechanisms to store and search measurements with time-series.

Fig. 3.6 shows the response time against different number of users for Query1 and Query2. The response time of EKB-TS is longer than EKB-TSDB. The performance gap is bigger than the gap observed in the processing time. The response time of EKB-TSDB is nearly two orders of magnitude faster. In addition to better data searching performance, EKB-TSDB also shows improved capability of data transmission. The reason is mainly the use of different data structures in the two systems: key-value for TSDB and triple for the triple store. The data size of a reply returned by the triple store is larger than that of TSDB, even if the content of information is the same. The triple store needs redundant triples to describe the format of measurement data. In addition, the response time of both implementations rises with increasing numbers of users, but EKB-TSDB scales better than EKB-TS when multiple users query data at same time.

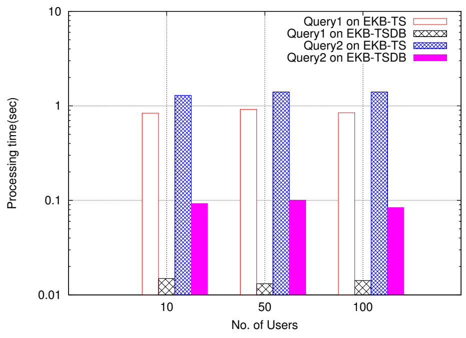


Figure 3.5: Processing time against the number of clients on two EKB implementations – EKB-TS and EKB-TSDB.

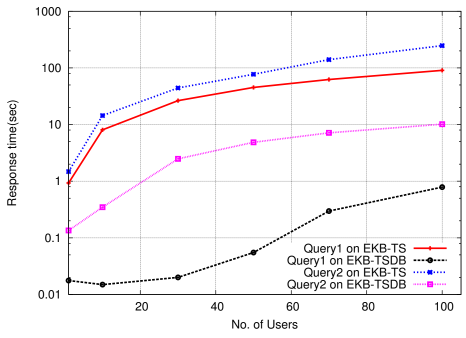


Figure 3.6: Response time against the number of clients on two EKB implementations – EKB-TS and EKB-TSDB.

The performance of Query3 for EKB-TSDB and EKB-TS is the same, below 0.01 second. We don't show their performance in the figures. From the two experiments above, EKB-TSDB enhances the efficiency of data communication and sharing between users. But there is a notable feature in the design of EKB: the server in the QMA, which deals with the request of users, plays a larger role than expected. In future implementation, the QMA could be duplicated to distribute the web queries if more than 100 users are expected to query one EKB agent.

3.6 Conclusion

In this chapter, we presented the Energy Knowledge Base (EKB) system which addresses the challenges of processing energy knowledge from heterogeneous resources across different administrative domains in the cases where information interoperability and scalability are important requirements.

EKB leverages EDL to instantiate data, which make data interoperable. EKB has validated the EDL ontology by using EDL in its knowledge components. The knowledge in EKB contains measurement data and metadata of e-Infrastructures.

We explained that they are both necessary for energy management. We provided two usecases for EKB to show its power in energy management.

We presented an implementation of EKB that stores the measurement data in TSDB without sacrificing its interoperability. The results of our experiments show that this implementation has better performance and can scale when numbers of users increase to 100.

Chapter 4

Evaluation of Power Estimation Models in a Computing Cluster

This chapter is based on:

- **H. Zhu**, P. Grosso, X. Liao and C. de Laat (2014). Evaluation of approaches for power estimation in a computing cluster. In 2014 International Green Computing Conference (IGCC), pages 1-10, Dallas, TX.
- **H. Zhu**, X. Liao, C. de Laat and P. Grosso (2015). Evaluation of non-linear approaches for power estimation in a computing cluster. Sustainable Computing: Informatics and Systems. (invited paper, under revision)

4.1 Introduction

Modern data centers use *workload consolidation* to improve the energy efficiency of a computing infrastructure [120][51][83]. Workload consolidation migrates workloads or virtual machines (VMs) from a set of current physical machine (PM) servers to a smaller set of servers. Data centers are usually under used, and rarely work at maximum performance [43]. Workload consolidation can achieve high energy efficiency by switching off unused nodes and increasing overall usage of resources.

In order to minimize energy consumption while meeting performance objectives when dynamically scheduling workloads, workload consolidation requires understanding of various cost factors. The most important factors are prediction of incoming workloads on a server as well as estimation of the energy consumption of VMs or the server as a result of the workload schedule to be carried out[1]. Considering that energy consumption equals to power consumption multiplied by time, it is necessary to estimate power consumption. *Power consumption estimation* computes the power according to current resource states from the Operating System (OS) or Performance Monitoring Counters (PMCs). The majority of power

[1]We don't estimate the power consumption of VMs in this chapter. There is no VM migration between PMs. In this case, workload consolidation can be performed through task migration – moving independent tasks between PMs, or killing tasks in current PM and setting up them on a different machine [58].

Table 4.1: Correlation coefficient of the resource features and power consumption obtained by measurement data of one typical node in the DAS-4 cluster. Part of feature types include 2 features due to two disks and two Network interface cards (NICs) equipped in one node.

Type of resource features	Corr(feature, power consumption)
CPU usage	0.89
Memory usage	0.79
Disk IOtime	0.48 (sda), 0.49 (sdb)
Disk read speed	0.46 (sda), 0.46 (sdb)
Disk write speed	0.38 (sda), 0.38 (sdb)
NIC incoming speed	0.37 (eth0), 0.36 (eth1)
NIC outgoing speed	0.38 (eth0), 0.18 (eth1)
ALL page fault	0.28
Major page fault	0.27

estimation models used before are linear models [42][134], namely linear functions of resource features such as CPU, memory or disk load.

From the measurement data from the Energy Knowledge Base for our cluster, we found that power consumption did not have a fully linear relationship with resource feature, in particular, I/O load. Table 4.1 shows a few types of resource features measured in the DAS-4 cluster in the left column and the correlation coefficient of each feature and power consumption in the right column. The correlation coefficient of CPU usage and power consumption is close to 1, while the coefficient for other resource features such as disk load and network traffic load is small. In fact, I/O resources show non-linear behaviour, e.g. disk I/O operations have the startup or stop delay of disk plates and the seek time of disk heads. Previous work also proved that power consumption was not fully linear with CPU or I/O load [62][52][90]. Therefore, non-linear approaches present an opportunity to achieve better accuracy than linear, only using the basic resource features of CPU, memory, disk and NIC.

Polynomial models and classification models, which are two common non-linear approaches in the machine learning area, are suitable for power estimation. Polynomial models can capture the distinct change rate of power consumption with respect to resource loads while classification models can describe power consumption as a function of resource features at different levels. We propose multiple non-linear approaches to train these models, and evaluate these models by comparing them with highly-used linear models. These power models provide the basis for controlling resources state for future scheduling. To pursue high usability in the schedule, our approaches only select basic controllable resource states in the OS as their features.

The accuracy of trained power models depends on the executed workloads. We trained power models for servers on the data collected from EKB with long-term previous runs rather than subjective benchmarking in a cluster. We consider three main reasons: First, there is no specialized benchmark for stressing all main server components. Existing benchmarks are mainly CPU or memory intensive but there is no standardized benchmark for evaluating hard disk or network resources. Second, benchmarks are designed to stress some of the server components but they couldn't represent the actual workloads in a cluster. It's reasonable to train models using actual workloads in the cluster. Third, the effect of machine learning methods depends on the size of the training set. Therefore, we collected a large volume of measurement data for over 3 months from our cluster to train models.

The power models are evaluated on each selected node with actual workload and some benchmarks. We measure the accuracy of the models on Hadoop nodes and on nodes with GPUs, evaluate the portability of the models among homogeneous nodes and analyse the usability of the models by comparing execution time, CPU usage and the size of training set needed.

The structure of this chapter is as follows: Sec. 4.2 presents related work on power estimation. Sec. 4.3 focuses on characterization of cluster workloads. Sec. 4.4 shows approaches for building power models. Sec. 4.5 provides the result of evaluation on the models. Finally, Sec. 4.6 shows conclusions.

4.2 Related Work

We differentiate between the concepts of *power consumption estimation* and *power consumption prediction*. Power prediction computes power consumption of nodes in future periods according to historical power consumption. Power consumption estimation contains the models for CPU, VM and server [103]. The models for VM and server are similar in terms of training approaches and features. The power consumption of a PM server can be calculated directly or as the sum of the power consumption of hosted VMs. It is therefore appropriate to discuss the state of the art about the power estimation of VMs and PM servers.

Many previous approaches have used linear models for estimating power consumption of a whole server. Fan et al. [69] implemented a linear model based on CPU usage only. As Rivoire et al. [114] proposed, the CPU usage only linear model is not an accurate reflection of the CPU power, and is only suitable for CPU-intensive workloads.

Models that use both OS-reported component utilization and CPU performance counters are increasingly necessary for accurate power estimation. Heath et al. [77] proposed a linear model for a heterogeneous server cluster. The power consumption of individual servers is estimated with a linear model that solely employs utilization metrics. The key factor is to determine the utilization level of each resource for a single server. The total consumption of a resource is the sum of the fraction of requests served locally, the cost of sending requests to other nodes and the cost of serving requests on behalf of other nodes. The utilization of a resource is derived by the ratio of resource consumption to capacity. Zamani et al. [133] created linear models using different combination of features such as OS-layer resource features, PMCs or various tasks execution cycles. Arbor et al. [41] analysed the correlation coefficient of PMCs and power consumption. They select five events with the highest correlation coefficients in their system-level power model, which is derived by multi-variable linear regression. Cache contention is considered as well. Economous et al. [65] also presented a full-system linear model, which considered the OS-reported features of main resource components such as CPU usage, memory usage, disk I/O rate and network I/O rate. The multiple-variable linear approach evaluated in this chapter uses similar features. All approaches above only generate a single linear model for one server. The accuracy of a single linear power model is limited for a variety of workloads.

The researchers address this challenge by characterizing workloads and generating a set of linear models for a server according to the workloads. A model is selected for power consumption estimation according to running workloads. Rajamani et al. [112] created linear models for CPU differentiated by value of parameters according to CPU operating frequencies and voltages. Li et al. [92] focused on the application of a linear power model in a Cloud environment. They observed low accuracy at the peaks and valleys of power consumption. The improved linear model is divided into 3 subclass models according to high, medium and low level of CPU utilization. Kansal et al. [83] learned a set of parameters for a model of each VM separately, based on the initial observation. The parameters for the VM can be shared if the hosting server has the similar configuration. But the methods of characterizing workloads in these literatures are not systemic, and are based on observations of a limited number of workloads. These methods are only feasible for known or observable workloads with obvious patterns.

Besides the common features from OS and PMCs mentioned before, few studies employed novel features in their models. Dhiman et al. [61] proposed a feature, which is ratio of instruction execution cycles to overall cycles, to characterize the workload. They created an Application Power Table of each application for each server type hosting the application. Parameters of the VM power model can be obtained from the table. Koller et al. [85] introduced a throughput-based power model according to observations from a set of experiments on a mix of diverse applications.

A new trend is to estimate power consumption through non-linear approaches. Lien et al. [93] suggested that a linear power model was only acceptable up to a medium utilization level. They built an exponential model for a web server, which is more accurate for different degrees of utilization. Lent [90] added a logistic function to model non-linear disk and OS behavior in a linear model for web servers. Wabmann et al. [127] computed power consumption of each VM in the server using a multiple-variable polynomial approach. Their features are the utilization of CPU, hard disk and NIC in the VM. Piga et al. [110] proposed a k-means model using a correlation-based feature selection. Zamani et al. [133] considered a time-series factor in their model. Dhiman et al. [62] used the Gaussian Mixture Models (GMM) to estimate the power consumption of each VM. The features they consider are IPC, MPC and CPU usage. They manually divide the CPU usage into many groups, and then cluster and fit each measurement vector using quantizer mismatch (QM) distortion in each group. After that, multiple Gauss clusters are generated inside each group. When estimating the corresponding power consumption for a given feature vector, the closest cluster to this vector can be calculated using QM distortion. The estimated power consumption is the power value in the center of the cluster. We implement a different GMM regression approach. We employ the Expectation-Maximization (EM) [104] algorithm to find clusters, and use conditional probability to estimate power consumption. Their GMM obtains the same approximate estimation value for all feature vectors fit in the same cluster while our approach is more fine-grained. Our approach calculates the estimation value for different feature vectors, even which are in the same cluster, using Bayesian theory.

Table 4.2: Summary of power estimation models for VMs and PMs; The default *accuracy* is on average if not specified

Literature	Platform	Feature	Approach	Benchmark	Accuracy
Fan et al. [69]	PM	CPU utilization	linear	GMail; Google Search	<1%
Rivoire et al. [114]	PM	CPU utilization; PMCs	linear	SPECcpu; SPECjbb; stream	8% (SPECfp); 9.5% (SPECint); 1.5% (stream)
Heath et al. [77]	PM	CPU utilization; network bandwidth; disk bandwidth; maximum number of concurrently open sockets	linear	mirco benchmark	1.3% avg; 2.7% max
Arbor et a.. [41]	PM	PMCs; Cache contention	linear	SPEC CPU2000	2.54% (avg); 4.14% (max)
Economou et al. [65]	PM	CPU utilization; memory accesses; disk I/O rate; network I/O rate	linear	SPEC CPU2000 ; stream	[0%,15%]
Kansal et al. [83]	VM	CPU utilization; LLCM	a set of linear	SPEC CPU2006	[1.6,1.8] Watts
Koller et al. [85]	PM	throughput; virtualization ratio	a set of linear	TPCW; SPEC Power; HPL	<5%; <5 Watts (mixed)
Lent et al. [90]	PM	the utilization of CPU, disk, NIC; memory access	logistic	web request	<5%
Dhiman et al. [62]	VM	CPU utilization; instructions; memory accesses; cache transactions	Gaussian Mixture Model	SPEC CPU2000	[8%,11%]; [1%,17%] abs
Wabmann et al. [127]	VM	the utilization of CPU, disk, NIC	polynomial	synthetic CPU workload	<4%

Table 4.2 summarizes the power consumption models for VMs and PMs we describe above. We can see there is no work to compare the effect of common non-linear approaches together, and most work only focus on the accuracy.

4.3 Load Measurement and Workload Characterization

Our power estimation study relies on measurement data, so we briefly introduce the collected data sets from our cluster.

4.3.1 Load and Power Measurement

The DAS-4 cluster system [8] is mainly used for research purposes, such as to process and test students' homework, benchmark mathematical models and validate research ideas. We only targeted single nodes with independent PDUs. We categorized 15 target nodes into two groups according to their use as shown in Table 4.3. One group is 7 *nodes with GPUs* which mainly run CPU intensive high performance computing; another is 8 *Hadoop nodes* which run short tasks that are assumed to be both CPU intensive and I/O intensive. We also run different types of benchmarks on the two groups for evaluation. The first group includes 7 nodes with K20m ”Kepler GPUs” and one of nodes with an extra Intel Xeon Phi accelerator. They are equipped with dual Intel "Sandy Bridge" E5-2620 (2.0 GHz) processors. All of the Hadoop nodes are homogeneous, equipped with the same processors as the nodes with GPUs, but Hadoop nodes are hyperthreading enabled. All the nodes in the cluster are homogeneous in memory, disks and NICs.

The OS is CentOS 6.2 x86_64. The current governor of Dynamic Voltage Frequency Scale (DVFS) capability is *userspace*, and the CPU frequency increases or decreases frequency within 1.20 GHz and 2.00 GHz depending on the set of userspace programs.

Table 4.3: The configuration of target nodes (two groups) in our cluster.

Group	Node with GPUs	Hadoop node
# of nodes	7	8
Processor	E5-2620 (2.0GHz), 12 cores	E5-2620 (2.0GHz), 12 cores, hyperthreading enabled
Memory	64GB	64GB
Storage	2*1TB	2*1TB
NIC	IB and GbE	IB and GbE
GPU	7 K20m, 1 Phi	none

The measurement data we used is stored in an Energy Knowledge Base agent belong to the DAS-4 cluster in VU University Amsterdam. The types of resource feature EKB can measure are listed in the Table 4.1. We generated one data set for each node. Each data set includes the measurement data for 100 days of the year 2013. The data is sampled every 3 minutes. Each data vector in a data set, which is an observation of different features and power consumption at the same time stamp, is called one example in machine learning terminology. Each data set contains about $100 * 24 * 60/3 = 48000$ examples.

4.3.2 Workload Characterization

The accuracy of power estimation approaches in a cluster depends on its workloads according to previous studies [90][63]. The approaches exhibit different effects in an I/O or CPU intensive environment. Some approaches are effective in an environment with mild workload pattern fluctuation, and are not suitable in a drastically fluctuating environments. For this reason, we characterize the workloads to understand our cluster environment before building estimation models.

According to the measurements data, the nodes with GPUs have a wide power range from around 130 to 330 watts while Hadoop nodes only change at most 100 watts between 110 and 210 watts.

In order to clearly describe the distribution of nodes in terms of main resource features, we compute separate normalizations for each resource field. The normalization is a scaling relative to the largest capacity of resources on any machine. This normalization method is also used in Google cluster data traces [12]. In Fig. 4.1 we show the CDF distribution of CPU, memory and disk load for all the nodes. The resource loads are CPU usage, memory usage and disk I/O time respectively. For the most of the time the CPU usage is close to idle; this is expected as our cluster is not a production system. Nodes with GPUs run few I/O tasks. Hadoop nodes run more I/O intensive tasks than nodes with GPUs because Hadoop nodes have higher memory and disk load.

To understand the fluctuation of our workload pattern, we visualize the load change of each node in our cluster. We compare the load change of our cluster with Google clusters, because they are known to exhibit strong workload variations. We

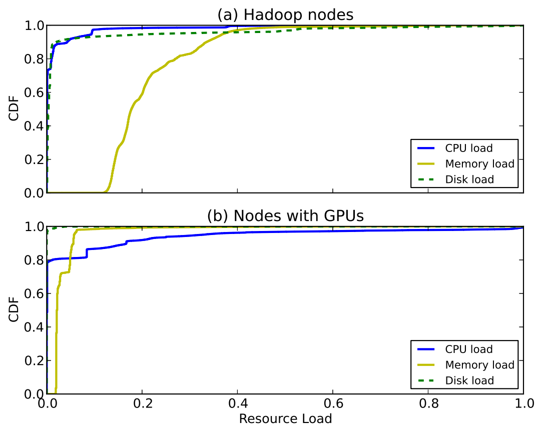


Figure 4.1: The CDF distribution of CPU, memory and disk load in our cluster: nodes with GPUs vs. Hadoop nodes

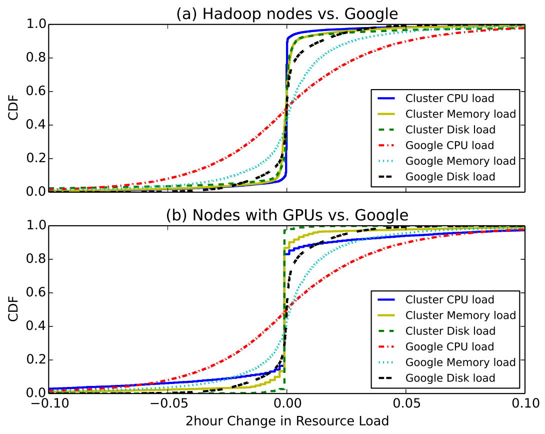


Figure 4.2: Changes in average CPU, memory and disk load between consecutive 2-hour sampling periods: nodes with GPUs vs. Hadoop nodes vs. Google

compute the average load change of every 2-hours for each node respectively, and then draw the curves of changes between two consecutive periods in Fig. 4.2. The Google nodes have indeed more drastic load change than our Hadoop nodes and nodes with GPUs, because Google Cloud runs many short tasks, which are abundant in task types. Our results of observing Google clusters are similar to what [113] did. This shows that our cluster tends to run workloads with mild load fluctuation. The load in the Hadoop nodes exhibits higher noise than that in the nodes with GPUs, because the Hadoop nodes tend to run shorter-term tasks compared with the nodes with GPUs. This further proves that the Hadoop nodes are more I/O intensive than the nodes with GPUs.

We conclude that the Hadoop nodes are in an environment with CPU and I/O intensive mixed workload while the nodes with GPUs mainly run CPU-intensive workloads.

4.4 Power Estimation Approaches

In this section, we describe the linear and non-linear power estimation approaches to be evaluated. All the approaches are based on the common features. We first discuss the selection of features from all the measured features.

4.4.1 Feature Selection

As seen in Table 4.1, we can obtain measurement data with nine types of resource feature. Two rules are followed to select the features for power estimation.

- First, we select at least one feature for each resource component. A server is composed of a few resource components; the main components are CPU, memory, storage and NIC.
- Second, we choose basic controllable features in the OS to ensure high model usability. Power models are the basis for controlling the state of resources for the schedule in the future. So even if the power consumption is estimated using the complicated features like CPU hardware counters, the OS couldn't precisely and easily control the state of these features. The scheduling decision could be impossible to execute.

According to Rule 2, we exclude *page fault*. We calculate the correlation coefficient between *Disk IOtime* and *Disk Speed* based on the data. The coefficient is very high (more than 0.88). This means they are effectively same feature for disks. We exclude *Disk IOtime* also. According to Rule 1 we select six types of feature: *CPU usage, memory usage, Disk write/read speed and NIC incoming/outgoing speed*. Effectively, we have 10 features due to two disks and two NICs equipped in each node in our cluster.

4.4.2 Approach Description

The analysis of Table 4.1 and previous work in Sec. 4.2 shows power consumption has a non-linear relationship with resource features. Besides linear approaches, we propose and implement two neural network approaches and one unsupervised classification approach to capture this relationship. The change rate of power consumption is distinct for different ranges of resource loads. For instance, power consumption grows slowly as resource loads increase from idle [52] or resources meet the bandwidth bottleneck for intensive workloads [90]; power consumption could take off under medium resource loads. The polynomial function in the neural network models can describe the non-uniform change rate of power consumption. A single set of parameters generated by linear or polynomial regression could not fully capture the complicated relationship between resource features and power consumption. This follows an observation from [62] – power consumption is a function of resource features at different levels. The relationship could be better represented using multiple sets of parameters generated by unsupervised classification models.

4.4.2.1 Linear and Multiple-variable Linear (M-Linear)

We implement two linear approaches; they both use an gradient descent algorithm to find the local minimum. One approach is a function of the CPU-only, named **Linear**, which was used in early power estimation studies. Another is a multiple-variable linear regression with the 10 features we selected, named by **M-Linear**, which is a direct optimization of the CPU-only approach.

4.4.2.2 Artificial Neural Network (ANN)

A feed forward neural network is one of the typical artificial neural networks, named **ANN**. Data flow always moves in one direction to calculate weights for each neuron. ANN adjusts the weights using gradient descent. Our ANN uses 2-degree polynomial functions in the neuron. The structure of our ANN network is predefined. The ANN network structure has 10 input variables in the input layer and 1 output variable in the output layer, because we input 10 features and only output power consumption. Based on our research experience on neural networks, we define a single hidden layer but multiple neurons in that layer.

4.4.2.3 Group Method of Data Handling (GMDH)

GMDH [108] is also known as a polynomial neural network, which can be represented as a set of neurons in which different pairs in each layer are connected through a polynomial function, thus producing new neurons in the next layer. The structure of a GMDH network is not predefined but evolves during the estimation process. Fig 4.3 shows an example of GMDH network with 4 inputs and 1 output. The number of input variables in each neuron is 2 and the maximum number of neurons in hidden layers is 3.

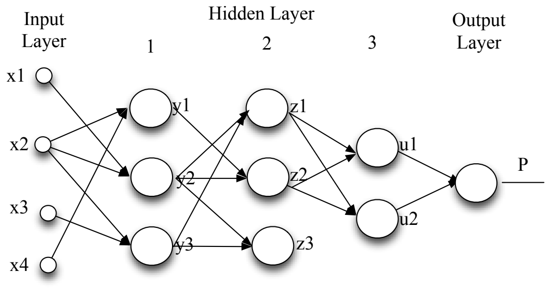


Figure 4.3: An example of Group Method of Data Handling network structure

Similar with ANN, GMDH use all 10 features as input variables and power consumption as an output variable. Our GMDH network training algorithm proceeds as follows:

1. Separating one data set into training set and validation set.
2. Creating combinations of the m input variables in each layer. In the first hidden layer, we input 10 features, so we create $\frac{10!}{m!(10-m)}$ combinations.

3. Calculating the regression for each combination. Each neuron is a d degree polynomial that captures the relationship between power consumption and the combinations. We estimate the weights of the neuron by the least square method for each combination. After regression, we compute the estimation error in the training set.
4. Selecting the intermediate neurons. Select the maximum n best neurons from all combinations according to the error. All the error of the selected neurons should be less than the largest error in the current layer. These n intermediate neurons are set as input variables of the next layer.
5. Computing the error of the selected neurons in the validation set. When the least error in the validation set for neurons in the next layer stops decreasing if compared with the least error in the training set in the current layer, stop training. Otherwise, jump back to step *2* to construct the next layer.

4.4.2.4 Gaussian Mixture Model (GMM)

GMM [50] is a parametric probability density function represented as a weighted sum of Gaussian cluster densities. Different from the two neural networks we discussed above, GMM is an unsupervised method, usually used for information classification. One of input parameters is the number of Gaussian clusters, denoted by c. The classification process is to find the parameters about each cluster, including cluster center, cluster co-variances matrices and mixing coefficients.

We leverage the GMM algorithm for regression because other classic classification methods such as K-means are not accurate for regression [121]. The process of GMM regression is as follows. First, 10 features together with power consumption are all used as input variables during training. Second, we use iterative EM to find the model parameters. Third, given the value of one feature vector, this feature vector's conditional probability density function can be obtained based on the model parameters [46]. So the cluster center, cluster co-variance matrices and mixing coefficients of the conditional probability density function are known. Finally, the power consumption for this given feature vector can be calculated as the expected value of the conditional probability density function, which is the sum of the mix coefficient for each cluster multiplied by the center of the cluster.

4.4.3 Methods of Training Models

We train the models for each node using the **Linear, M-Linear, ANN, GMDH and GMM** approaches. Before training, we classify each 100-day data set into three groups. The **Training set (TrS)** is used to fit the models and find the model parameters, for instance, for computing the weights for each neuron in GMDH and ANN, cluster parameters in GMM and weights for each feature in the linear models. The **Validation set (VS)** is used to decide the proper input parameters. For one set of input parameters for an approach, we obtain one model by training. We can get a couple of models for the same approach because of different values of input parameter. We choose the model with the optimal input parameters, which causes the least root mean square error in the validation set. The input parameters and their optimal values for typical nodes with GPUs and Hadoop nodes are shown in Table 4.4. VS can also prevent overfit of models.

In cases where training is performed for too long or where examples in the TrS are rare, overfit may take place. The accuracy in the TrS increases while the accuracy in unseen data set becomes worse. We can judge whether overfit happens by observing the accuracy in validation set. The **Test set (TeS)** is used to evaluate the effect of the chosen model. We set the ratio of TrS size, VS size and TeS size as 6:2:2. The size is the number of examples in the each set.

Table 4.4: Input parameters in the approaches and their optimal value for typical nodes with GPUs and Hadoop nodes

Approach	Notation	Definition	Node with GPU	Hadoop node
ANN	n	number of neurons in each hidden layer	14	6
GMDH	v	number of input variables for each neuron	11	11
	d	degree of polynomial function	2	3
	n	maximum number of neurons in the hidden layer	10	10
GMM	c	number of clusters	20	20
Linear	λ	regularization coefficient	0.9	0.9
	ϵ	learning rate	0.1	0.1
M-Linear	λ	regularization coefficient	0.7	0.9
	ϵ	learning rate	1	0.01

We use a random data set classification method to mitigate the locality of workloads. If the TrS and VS have widely different workload patterns, this could influence the choice of models using the VS and finally impact the evaluation result in the TeS. We shuffle all the examples in each initial data set according to time stamps. All the examples are random in the time sequence. From an overall data set, we allocate the first 60%, the following 20% and the remaining 20% of all the examples into the TrS, VS and TeS respectively. We shuffle and classify the data sets 50 times. For one shuffle, we train, select and evaluate the model one time. All the evaluation results we show in the next section are the mean value of the 50 times.

4.5 Evaluation

4.5.1 Metrics for Accuracy

We introduce two metrics to evaluate the accuracy of estimation: Root Mean Square Error (RMSE) and success ratio. For one node, RMSE measures error between the estimated power and the measured power in its test set. The size of examples in the estimated data set is M. We denote the m_{th} real power consumption of node i in this set by $P^i(m)$. The RMSE of one node i can be represented as:

$$RMSE(i) = \sqrt{\frac{1}{M}\sum_{m=1}^{M}(p^i(m) - P^i(m))} \tag{4.1}$$

In addition, we compute success ratio, which is the ratio of the number of accurate estimations to the total number of estimations. An estimation is deemed accurate if it falls within some Δ of the real value. In our case, Δ can be 1%, 5% or 10%. The formula for computing the success ratio of one node i is shown as:

$$\begin{aligned} SR(i) &= \frac{\sum_{m=1}^{M} g(m)}{M} \\ g(m) &= \begin{cases} 1 & \text{if } \dfrac{p^i(m) - P^i(m)}{P^i(m)} \leq \Delta \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.2}$$

Success ratio and RMSE are two different metrics. RMSE measures the mean performance of multiple estimations, while success ratio depicts the distribution of accurate estimations. If the estimations have high error very few times and most estimations are precise, they may show high RMSE, but also a high success ratio.

4.5.2 Accuracy Results

We compare RMSE and success ratio of all nodes on the originally shuffled test set, and then we analyse the results on nodes with GPUs and on Hadoop nodes respectively for various benchmarks.

4.5.2.1 In The Test Set

We can observe the result of each node from the Fig. 4.4 and 4.5. Fig. 4.4 depicts the CDF distribution of RMSE of all the nodes. GMM and GMDH show the lowest RMSE for all the nodes, and have good estimation effect. ANN has the similar RMSE error with the M-Linear model. The Linear model has slightly higher RMSE than the M-Linear model for Hadoop nodes. For these Hadoop nodes, the CPU is not the only component that dominates the power consumption of the whole node considering that many I/O intensive tasks are run.

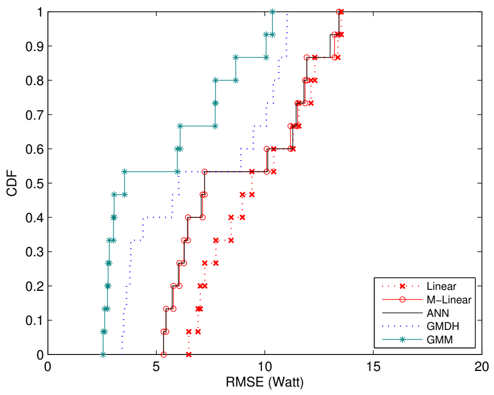


Figure 4.4: The CDF distribution of RMSE of all the nodes in the test set

Fig. 4.5 is the CDF distribution of success ratios for all the nodes with Δ 1%, 5% and 10%. Success ratio goes up with the increase of Δ. Although the Linear model has similar RMSE performance with M-Linear and ANN for some nodes, the success ratio of the Linear model is lower than that of M-Linear and ANN for all the nodes. The success ratio of ANN is nearly identical with that of M-Linear. GMM has the best result when Δ is 1%, and the gap between GMM and GMDH becomes narrow when Δ increases. When we set Δ to 10% in Fig. 4.5(c), the success ratio of all the models is more than 90%. This is to say less than 10% of estimations have the ratio of estimation error to the real power consumption over 10%.

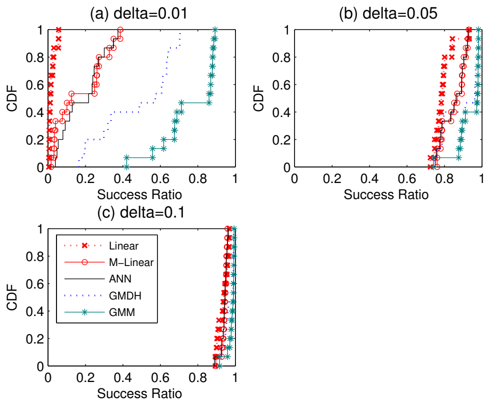


Figure 4.5: CDF of success ratios of all the nodes in ranges of the Δ 1% (a), 5% (b) and 10% (c) in the original test set

In the test set, we conclude that GMM and GMDH perform better than other approaches, because they capture more complex data dynamics and key feature factors. Among neural networks, GMDH is better than the ANN approach. The reason is that the GMDH network structure is resolved by means of an adaptive synthesis during the estimation process while ANN is simply predefined. The structure of ANN depends on experience. Moreover, it is difficult with ANN to guarantee global convergence. Simple-structure ANN has no advantage over M-Linear. M-linear is better than the Linear approach because it captures I/O features.The five approaches exhibit high RMSE and success ratio accuracy when training and testing models using actual workloads. Next, we test the approaches for some benchmarks, which probably have different patterns than with actual workloads.

4.5.2.2 For The Benchmarks

We further study the estimation performance for two types of benchmark. The benchmark information is shown in Table 4.5. We run Linpack on the four nodes only with a single GPU and we run MapReduce benchmarks on the eight Hadoop nodes. *MapReduce Writer* is a low CPU usage but I/O intensive benchmark, which

Table 4.5: Benchmarks and target resources for evaluation

Benchmark	Parameters	CPU load	Target resources
MapReduce Writer	20G data per map; 10 maps per node	5%-20%	8 Hadoop nodes
MapReduce Sort	200G per node; 10 maps/reduces per node	75%-100%	8 Hadoop nodes
Linpack	N: 40000; P*Q: 12, 24, 36, 48	25%, 50%, 75%, 100%	4 nodes with GPUs
Linpack-GPU	N: 40000; P*Q: 24	10% - 20%	4 K20 GPUs

writes random keys and values into a big HDFS file. *MapReduce Sort* is a CPU-intensive benchmark that sorts the data generated by the Writer benchmark. The CPU usage changes in a range of 5%-20% and 75%-100% when running Writer and Sort respectively by the parameters we specify. *Linpack* is a CPU stress benchmark. Our Linpack test set has a problem size of 40000 and it includes 4 Linpack tests. We adjust the process grid ratio ($P * Q$) to change in ranges of 12, 24, 36 and 48 in our cluster, which makes CPU load change in ranges of 25%, 50%, 75% and 100% on each node. *Linpack-GPU* is a same benchmark as the Linpack test with the process grid ratio of 24, but this benchmark is executed on GPUs rather than CPUs.

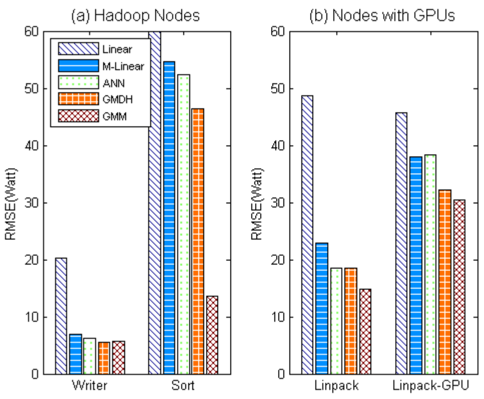


Figure 4.6: Mean RMSE of nodes for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort. (b) The Linpack test and Linpack-GPU test are both with a process grid ratio of 24.

The Fig 4.6(a) shows mean RMSE for the MapReduce benchmark on the Hadoop nodes. For Writer, all the models achieve high accuracy except the Linear model. The mean RMSE for Sort is much higher than for Writer for all the models. This means the estimation error for CPU intensive tasks is higher than that for

I/O intensive tasks. GMM gives a remarkable improvement over the other models, even in high CPU usage situations on the Hadoop nodes. The mean RMSE of GMM is at least half that of the other models. The ANN model gives a small enhancement for RMSE, only about 4% less than M-Linear. GMDH is 15% better than M-Linear. M-Linear obviously outperforms Linear.

Fig 4.6(b) shows the mean RMSE for the Linpack test and for the Linpack-GPU test both with a process grid ratio of 24. The two neural networks offer a slight improvement over M-Linear. Compared with the results under Linpack and Linpack-GPU, power consumption is estimated to be worse when running tasks on the GPUs.

Fig. 4.7 presents mean success ratio on the Hadoop nodes and on the nodes with GPUs under different Δ. It shows that GMM has the best success ratio in most situations. In some cases models have lower RMSE error but less success ratio. For example, mean success ratio of GMDH is higher than GMM when Δ is 5% for the Writer benchmark. This is because some estimations of GMM and M-Linear have large error.

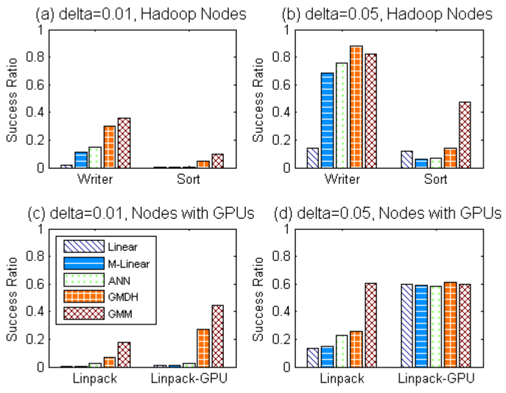


Figure 4.7: Mean success ratio of nodes for the benchmarks. (a)(b) The MapReduce benchmark includes Writer and Sort. (c)(d) The Linpack test and Linpack-GPU test are both with a process grid ratio of 24.

We evaluate mean RMSE for the Linpack test set which consumes different CPU usage on the nodes with GPUs. In Fig. 4.8, the mean RMSE basically increases when CPU usage rises for all the models. M-Linear is still much better than Linear. GMDH becomes less accurate than ANN with growing usage. They are both better than the two linear models. But GMM no longer remains the most accurate at high CPU usage for these CPU intensive nodes. This might be the result of an imprecise cluster. According to We propose a simple optimized GMM model (**GMM-Opt**) with the awareness of CPU usage. We manually classify each data set into 5 groups according to the CPU usage: [0, 0.2], [0.2, 0.4], [0.4, 0.6], [0.6, 0.8] and [0.8, 1]. Then we train models independently in each group. We fit each feature vector in a proper group based on its CPU usage value to determine a model, and finally compute the power estimations. The figure shows GMM-Opt

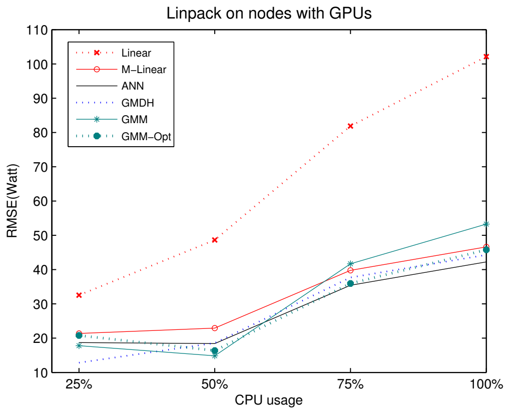


Figure 4.8: Mean RMSE of nodes with GPUs for Linpack. The Linpack tests consume CPU usage in ranges of 25%, 50%, 75% and 100%.

improves the accuracy of GMM and has a similar effect to the neural network models.

In summary, for the Hadoop nodes with an I/O and CPU-intensive mixed computing environment, GMM has the better accuracy than GMDH in particular for CPU-intensive workloads; For nodes with GPUs that are purely CPU-intensive, GMM can be optimized with manual clustering to achieve better performance. In both environments, ANN shows its improvement over M-Linear under resource intensive workloads, but improvement of the two neural network approaches over M-Linear is not remarkable. M-linear is much better than the Linear approach.

4.5.3 Portability and Usability Results

In a cluster environment, not all the nodes are installed with power meters. How to obtain their power models? Power is consumed by resource components to carry out tasks. Power consumption should be same for computer systems with homogeneous hardware and software configuration. Thus, models can be shared between homogeneous nodes. We have generated 5 different models using 5 approaches for each node. We apply the models for one node onto homogeneous nodes in the cluster to evaluate error in the test set. This is a process of testing the portability.

Fig. 4.9 shows the fit ratio of each model for the Linpack and MapReduce benchmarks. The MapReduce benchmark includes Writer and Sort. The fit ratio is the absolute error between the estimated power consumption using migrated models and the originally estimated power consumption, divided by the originally estimated power consumption. We traverse all the nodes to check the fit ratio for its homogeneous nodes. We list the best and worst fit ratio values. The two neural network approaches have the worst effect because the fit error could be enlarged by high degree polynomial function in neurons even with minimal difference from weights. GMM performs well among homogeneous nodes. M-Linear has the better portability than Linear for Linpack.

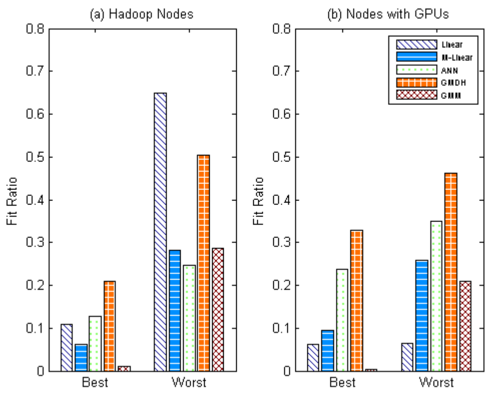


Figure 4.9: The best and worst fit ratio using model migration for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort; (b) Linpack is tested with a process grid ratio of 24

In practice, system operators aren't only concerned with the estimation accuracy of models. They may also consider the training time, estimation time and resource usage of each approach. In our last experiment, we study these factors. We train all the models and estimate power on a server node with an Intel E5620 processor (2.4GHz) and 24 GB memory. The range of results in the test set and for the benchmarks we use above are shown in Table 4.6.

The training time varies a lot. Both linear models need less than 10 seconds. The times of GMM and GMM-Opt are the longest, nearly 30-100 fold that of linear models. ANN and GMDH take about 60 seconds. All the models require less than 1 millisecond to make one estimation on average. The CPU usage for model training is nearly the same, less than 8%. In the process of training, we attempt to build models using a different size of TrS. We observe that the two linear models don't require a large amount of training data to reach a stable RMSE value, while non-linear models could suffer overfit with inaccurate results if the TrS is not large enough.

Table 4.6: The training and estimation time cost, resource usage and TrS demand of the approaches

	Linear	M-Linear	ANN	GMDH	GMM/ GMM-Opt
Training time (sec)	2-4	4-7	25-73	17-60	132-227
Estimation time of single example (sec)	$< 10^{-8}$	$< 10^{-7}$	10^{-7} - 10^{-6}	10^{-4} - 10^{-3}	10^{-4} - 10^{-3}
CPU load	<7%	<7%	<8%	<8%	<8%
Large TrS demand	No	No	Yes	Yes	Yes

Cluster operators can use the GMM or optimized GMM approach if they demand a precise model in a long schedule time window. They have enough time to train the model, estimate power consumption and perform schedule in the window. If the operators are concerned with computing costs in a short schedule window or in a hard deadline for scheduling, they should choose M-linear. Machine learning approaches take too much computation time. When they compute the models, the incoming short schedule window could have passed in the worst case. The simple approaches are adaptive in the environment where the configuration of nodes changes or updates frequently. The linear approaches allow operators to update their models in a short time, while the complicated machine learning approaches need quite a long time to collect new measurements for new model training.

4.6 Conclusion

In this chapter we described the importance of power estimation and the evidence of its non-linear characteristic, which motivated our evaluation of the most-used linear approaches as well as our proposed non-linear machine learning approaches including ANN, GMDH and GMM on the Hadoop nodes and on the nodes with GPUs. We evaluated the approaches in a cluster environment using a large amount of measurement data.

The non-linear machine learning approaches can improve on the estimation accuracy of the linear approaches using basic resource features. The GMM approach has the best power estimation accuracy compared with the neural network approaches and the linear approaches in an I/O and CPU-intensive mixed environment; GMM is not suitable in a purely CPU-intensive environment, but the optimized GMM approach can improve GMM and achieve similar performance compared to the two neural networks. The GMM regression consumes the longest time to train the model. The neural network approaches only have a slight accuracy advantage over multiple-variable linear approach; meanwhile these neural network approaches have the worse portability when they are applied to homogeneous nodes. A multiple-variable linear approach highly improves the estimation accuracy of CPU-only linear approach. We suggest to use GMM or optimized GMM if one is only concerned with estimation accuracy, and to use a multiple variable linear regression in time constrained environments.

Chapter 5

Joint Flow Routing-Scheduling for Energy Efficient Software Defined Data Center Networks

—A Prototype of an Energy-aware Network Management Platform Based on OpenNaaS

This chapter is based on:

• **H. Zhu**, J. Aznar Baranda, C. de Laat and P. Grosso (2015). Green routing for software defined data center networks based on OpenNaaS. 4th International Conference on Green IT Solutions (ICGreen), Milan, Italy.
• **H. Zhu**, X. Liao, C. de Laat and P. Grosso (2015). Joint flow routing-scheduling for energy efficient software defined data center networks. Journal of Network and Computer Applications. (under revision)

5.1 Introduction

In the total energy footprint of data centers, the proportion of energy consumed by the network components can be up to 50% if optimal power management techniques are used on the server-side [38]. This occurs particularly in data centers in an e-Infrastructure where large volumes of data are frequently transported. It's therefore crucial to reduce the energy consumption of networks in data centers.

Advanced network architectures in data centers such as Fat-tree [40] and BCube [74] are usually over-provisioned, with full-connected topologies and multi-path routing to guarantee large network capacity and high robustness. A large number of network resources are used to meet performance requirements at peak times. However, these resources are usually underused and rarely work at the peak performance. Unfortunately, networks in a data center are not power proportional – networks with low loads still consume more than 90% of the power used during the busiest hours [78]. They effectively suffer from inefficient power usage when traffic is not heavy.

Energy-aware routing techniques are effective approaches that can fix this problem. They are in essence strategies which focus on the energy state of the network, e.g. energy consumption or CO2 emission rate. They make routing decisions to aggregate traffic over a subset of links and devices in over-provisioned networks and switch off unused network components.

There are two ways to implement energy-aware routing: one is in IP networks, the other modality is to focus on data centers using Software Defined Networking (SDN). In IP networks we can for example observe the development of a Green OSPF protocol based on the OSPF protocol [53]; its energy-aware algorithm uses a subset of shortest path trees to select the routing path and activates as few as possible links to route traffic.

In SDN, in particular when looking at networks adopting OpenFlow [101], control plane (routing decision) decoupled from data plane (data forwarding) is moved to a centralized controller. Energy-aware routing could be easily implemented in the control plane. We work on data centers using Software Defined Networking in this chapter. The question is what the most effective manner is to achieve this.

Previous studies, which we will further discuss in Sec. 5.3, implemented energy-aware routing using various OpenFlow controllers, such as NOX [24] and Beacon [67]. These implementation has two drawbacks. First, neither implementation can easily be migrated between SDN networks with different types of SDN controllers because of incompatible structure and APIs. Second, energy-aware routing techniques rely on precise information about current topology and traffic, but the capabilities of the OpenFlow controllers are usually limited in their ability to obtain this information e.g. NOX can only discover the network topology, and most of the controller can't obtain topology or traffic statistics. Existing network management platforms can support multiple types of controller, and they have no compatibility problem when integrating new energy-optimizer modules into them. They can provide the capabilities missing from the controllers. Considering these factors as well as the extensive structure and powerful network management capabilities of OpenNaaS, we decided to concentrate on OpenNaaS as a suitable platform for the implementation of energy-aware services in SDN.

We used the features of OpenNaaS to provide the capability to monitor energy state and make energy-aware routing decisions. The Energy Language Description Energy is used to define data elements and their structure for energy monitoring. Monitoring capabilities pave the way for implementing energy-aware routing as well as other power management techniques in networks e.g. adaptive link rate (ALR) [73]. Green routing capabilities can calculate energy-aware routes and push the routes to OpenFlow switches. We implemented an initial prototype of Energy-aware OpenNaaS and validated its functionality using a greedy routing algorithm.

After that, we discuss a method to find a better energy-aware routing strategy for OpenNaaS. Rather than only targeting routing algorithms, we evaluated several routing strategies, namely we considered both how to find routing paths for flows and how to schedule the flows on the same link. We selected the best strategy by evaluating the energy consumption and mean flow completion time. Our simulation shows that this best strategy is a combination of priority-based shortest routing and exclusive flow scheduling, thanks to which we can achieve 5%-35% higher energy efficiency than common routing strategies without performance degradation when traffic consists of large-sized (5GB) of flows.

The structure of this chapter is as follows: Sec. 5.2 presents the energy-aware routing problem; we follow with a section detailing the related work on software management platforms and green network techniques (Sec. 5.3). Sec. 5.4 describes the OpenNaaS framework, while Sec. 5.5 introduces the design of energy-aware OpenNaaS. Sec. 5.6 provides a practical usecase of energy-aware OpenNaaS. After that, Sec. 5.7 discusses the design and selection of energy-aware strategies and Sec. 5.8 evaluates the strategies by simulation. Sec. 5.9 and Sec. 5.10 present discussion and conclusions.

5.2 The Energy-aware Routing Problem

The power consumption of a switch is the sum of static power and dynamic power, according to the power benchmarking results of various network devices in [99][139]. Static power is constant and it includes power consumed by chassis, fabric, fans, etc. Dynamic power is consumed by device interfaces and is related to the rate of traffic across the switch.

If we denote $P(u)$ as the power consumption of a switch, this will depend on the set of enabled interfaces and the load of each one of them:

$$P(u) = P_{base} + \sum_{j=1}^{N} a_j \cdot p_j(u_j) \tag{5.1}$$

$$u_j = \sum_{k=1}^{K} u_{j,k}, 0 \leqslant u_j \leqslant C \tag{5.2}$$

Where P_{base} is the static power; N is the number of links on the switch; $p_j(u)$ is the power consumption of a switch interface j at utilization u; a_j is the binary decision variable indicating whether the interface j is powered on. Given K traffic flows $f_1, f_2, ..., f_k, ..., f_K$, the utilization of f_k along the link j is $u_{j,k}$; while the capacity of the link is C.

Multipath routing algorithms [94], such as equal-cost multipath (ECMP), use multiple paths while sending flows, but do suffer from longer delay and additional control messages. In our study we assumed that no flow is split onto multiple paths, denoted by:

$$\forall k, \text{ if } u_{j',k} > 0, \sum_{j=1, j \neq j'}^{N} u_{j,k} = 0 \tag{5.3}$$

Energy-aware routing is an effective solution to save energy by aggregating the traffic over a subset of network links or network devices in over-provisioned networks. Solving the energy-aware routing problem is equivalent to minimizing energy consumption of a network:

$$\text{Min} \sum_{i=1}^{M} b^i \cdot P^i(u) \cdot t^i(u) \tag{5.4}$$

Where $P^i(u)$ denotes the power consumption of switch i at utilization u; M is the number of switches in the network; b^i is the binary decision variable indicating whether switch i is powered on; $t^i(u)$ denotes the time used for transferring the traffic by the switch i at utilization u.

Most previous energy-aware routing studies assume the completion time $t^i(u)$ is constant. In fact, they simplify the energy-aware routing problem as Eq. 5.5 to find a power-minimized network subset for traffic:

$$Min \sum_{i=1}^{M} b^i \cdot P^i(u) \tag{5.5}$$

It is obvious that in this case the solution to the minimization problem could be not energy efficient. Even so, finding the optimal flow routing for power-minimization is an NP-hard problem [78][131]. It is difficult to solve the power minimization problem using a formulation, in particular for large scale networks. The universal solutions are to compute a route for each flow and generate a subset of the network by combining all the result routes. We will discuss the state of the art for energy-aware routing next section.

5.3 Related Work

5.3.1 Energy-aware Routing

Cianfrani et al. [53] proposed a Green OSPF protocol. Its energy-aware algorithm only uses a subset of router Shortest Path Trees to select the routing, reducing the number of links used to route traffic. Bianzino et al. [45] designed a link-state protocol with a fully distributed solution to save on the power consumption of links. The above studies focus on the implementation of energy-aware routing in IP networks.

Existing work that implements energy-aware routing algorithms for data center networks (DCNs) using SDN is mostly theoretical, and only a few cases exist of actual implementations.

Several authors have tackled the problem theoretically. Wang et al. [128] analyzed the correlation between flows, and their correlation-aware routing algorithm greedily consolidates as many weak-correlation flows as possible onto a path. They further adapt the data rate of each link to save power as links may not be fully utilized in the routing algorithm. Power can be saved by migrating Virtual machines (VMs) to a smaller set of servers and powering off unused servers in data centers [39]. Zheng et al. [136] combined Wang's work with VM consolidation for further power saving.

Liu et al. [96] proposed a distributed strategy to find the routing path for elephant flows and make tradeoffs between energy consumption and network performance in a DCN. They observed that elephant flows easily lead to network congestion, so their elephant flow routing greedily selects the path with the least network utilization caused by the running elephant flows to improve network performance.

Xu et al. [131] [118] proposed a power-aware routing algorithm for reducing power consumption of high-density data center networks while meeting the overall throughput requirement. The idea of the algorithm is to compute a basic routing path set for all flows and then to remove switches and links from the path set without violating the demand of overall network throughput.

Fang et al. [70] proposed a similar routing scheme. The scheme first selects a minimal subset of network elements by using the Steiner tree framework, and then uses a multiple-routing algorithm to find multiple routes for each flow. Finally, it combines the routes that need to activate as few unselected elements as possible along with the initial subset to generate the final network subset.

Others have provided actual implementations and prototypes. Jin et al. [82] converted the VM placement problem into a routing problem, which combined hosts and network based power optimization. They use depth-first search to quickly traverse the hierarchical layers between VM pairs in a DCN, and employ a best-fit criterion in terms of memory demand of VMs to determine the link between any two layers. Their prototype integrated the depth-first best fit rule into the Beacon OpenFlow controller.

Heller et al. [78] presented ElasticTree for adapting the power usage in a Fat Tree data center with OpenFlow switches. ElasticTree uses NOX that is an original OpenFlow controller to pull traffic data and push computed flow routes to each switch; it employs an optimizer to compute power-minimization routes which meet current traffic conditions. Mahadevan et al. [98] combined a routing algorithm used in Elastic Tree and the algorithm of server load consolidation that migrates jobs to use fewer servers for data center networks. They found that 74% percent of total network power can be saved.

Thanh et al. [123] presented a platform to measure and analyze the power consumption of software-defined DCNs using energy-aware topology optimization and routing algorithm. The platform implemented the same routing algorithm with ElasticTree in NOX controllers based on the power profiling of NetFPGA switches. Thanh et al. [124] improved the previous routing algorithm to enable an adaptive link rate technique.

Our work is different from these previous work in terms of algorithms and implementation.

The studies above only focus on routing algorithms and evaluation of power consumption, and they neglect the scheduling of flows on the same link. Li et al. [91] studied a scheduling algorithm, however the analysis of routing algorithms is missing. The energy-aware routing strategies we proposed and evaluated include not only routing algorithms but also scheduling algorithms. In addition, we evaluated algorithms for original energy-aware routing problem concerning energy consumption (See Eq. 5.4) instead of the power-minimization problem (See Eq. 5.5).

Besides, we implemented a prototype based on a platform – OpenNaaS. Our energy optimizer component is implemented in this network management platform rather than only based on SDN controllers. Previous systems, for example ElasticTree, can't dynamically discover the topology and monitor the power consumption of switches due to the limitation of SDN controllers; they assume that the topology and power are always invariable once they are manually input. Energy-aware OpenNaaS inherits and enhances powerful network management capabilities, e.g. dynamically obtaining power and topology information, and supporting multiple types of SDN controllers, so it has wide applicability.

5.3.2 Cloud/Network Management Platform

Given that OpenNaaS is effectively a full network platform, it is fair to compare it with existing cloud/network management platforms that also support SDN technologies.

OpenNebula and OpenStack are both open-source cloud computing platforms for public and private clouds. Cloud operators can control computing, storage and network resource in clouds through their APIs. Their network capabilities are limited to IP, vLANs and SDN currently. OpenNaaS has more sufficient network capabilities e.g. Bandwidth on Demand (BoD), topology discovery, and Generic Routing Encapsulation(GRE) and support more usecases, e.g. virtual Customer Premises Equipment (vCPE) than OpenNebula and OpenStack. We can implement more energy-aware capabilities like energy-aware BoD by combining energy monitoring with existing network capabilities in OpenNaaS. Besides, OpenNaaS has a well-organized structure to enable the abstraction of underlying network technologies and resources, easily extended to implement new network technologies.

Nuage Networks [23] and the Tail-f Network Control System [28] both provide general SDN capabilities and vCPE solutions for data center networks. They offer a virtualization environment for a data center network, but these software-based products are not open source, and as such cannot be easily extended.

RouteFlow [105] provides remote IP routing services based on a set of open-source software. RouteFlow doesn't make a routing decision, which depends on a virtualized IP routing engine, Quagga, that provides the implementation of routing protocols e.g. OSPF and BGP. [106]. RouteFlow only focuses on routing services and its structure is not easily extensible for new network functionality.

5.4 OpenNaaS Framework

OpenNaaS [25] is the outcome of the European Community Mantychore FP7 project. It is as an open source platform for the GÉANT network operations center (NOC), NRENs and other infrastructure providers of network and computing resources, which has been proposed to provide NaaS-based services. The main contributors include Juniper, HEAnet and i2CAT. OpenNaaS is proposed as a common network management and service orchestration platform, capable of providing and managing network in a flexible and efficient way. OpenNaaS can take advantage of SDN and Network Functions Virtualization (NFV) technologies.

OpenNaaS abstracts aside physical resources, enabling physical topology and vendor-specific details to be decoupled from their control and management features. The fundamental unit that OpenNaaS uses to accomplish this is the *Resource*. A Resource models a device and represents a manageable unit inside the NaaS concept e.g., a switch, a router, a link, a logical router, or a network. The basic resource considered is the Physical Resource (PR); Virtual Resources (VRs) are then created by manipulating the PRs. *Capabilities* shape resource functionality and provide an interface onto given resource functionality, e.g. OSPF, IPv6, the ability to create/manage logical routers, etc for a router. Fig. 5.1 shows this OpenNaaS vision in which physical devices are abstracted into resources and capabilities whose management can be delegated to upper application layers. The OpenNaaS model aims to be flexible enough to support different designs and orientations, but fixed enough so common tools can be built and reused across plugins.

The implementation of OpenNaaS consists of two distinctive parts: the core and the extensions. The core can be understood as a provider of basic functionality, e.g. resource management, which can then be used by the extensions. The

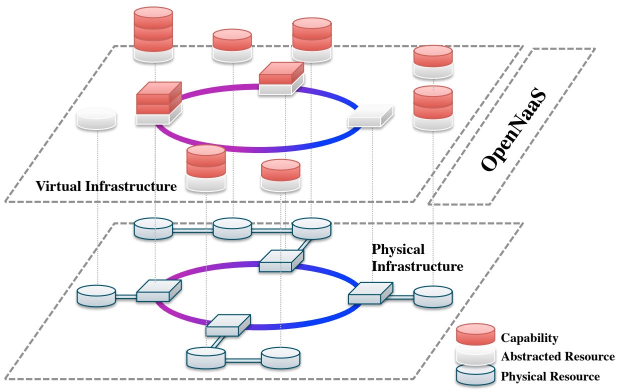


Figure 5.1: OpenNaaS abstraction view: resources and capabilities.

extensions provide functionality for a specific aspect of networking, e.g. configuration of routers, by defining capabilities the resources have. The structure of OpenNaaS allows developers to easily implement more functionality by creating capabilities in a new extension bundle. We exploited this feature to create an energy-aware bundle (see Sec. 5.5.2).

OpenNaaS has a complete view of the entire network and interacts directly with the data plane through its SDN capabilities. It separates the control and data planes and its architecture follows the SDN paradigm. The OpenNaaS framework is widely used as an enabler for SDN technologies. OpenNaaS interworks several SDN platforms (OpenDaylight [26], RYU [27] and Floodlight controllers [21]) to orchestrate network services on top of SDN-based infrastructures and to enable new SDN applications for different stakeholders.

OpenNaaS defines an OpenFlow resource model for OpenFlow switches and create capabilities for OpenFlow resources in the *OpenFlow bundle*. The OpenFlow bundle works as an OpenFlow driver, which accesses the REST APIs of OpenFlow controllers for port statistics and flow forwarding (allowing developers to create, remove and get forwarding rules).

5.5 Design of Energy-aware OpenNaaS

We integrated an energy-aware bundle in OpenNaaS to allow energy monitoring and green routing capabilities for OpenFlow networks.

5.5.1 Architecture

Fig. 5.2 shows the architecture of the energy-aware OpenNaaS we developed. The OpenNaaS server runs on top of an OpenFlow controller (OFC), and the OpenFlow-enabled switches are connected with the OFC. There are two methods to invoke green routing functionality in OpenNaaS: 1) A network provider or user directly sends a green routing request through scripts or GUI to OpenNaaS; 2) The OFC detects a packet-in event in a switch and then sends a routing request to

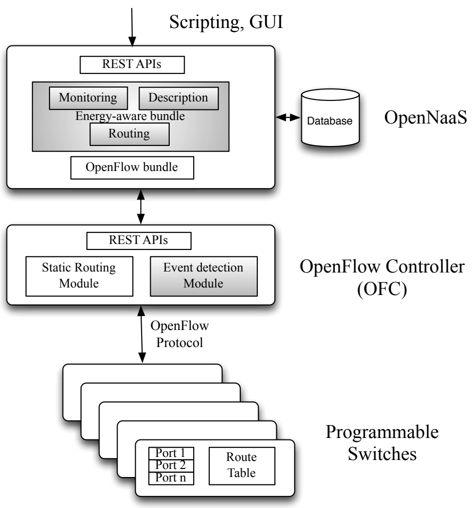


Figure 5.2: Energy-aware OpenNaaS architecture

the OpenNaaS server. The *energy-aware bundle* in OpenNaaS receives and handles the request. This bundle is capable of calculating a green route and making a routing decision. The energy-aware bundle, which communicates with the OFC through the *OpenFlow bundle* calls the defined REST APIs in the OFC to add static flow rules for the route. The *static routing module* in the OFC executes the add operations by inserting flow entries into the flow tables of the switches.

5.5.2 Energy-aware Bundle

The *Energy-aware bundle* at the core of our design implements energy-aware description, monitoring and routing capabilities for OpenFlow resources. Energy-aware routing depends on topology information, traffic information and energy usage information to determine the configuration of the network when scheduling traffic. OpenNaaS can provide the first two pieces of information. OpenNaaS employs the Link Layer Discovery Protocol (LLDP) to obtain network topology and uses OpenFlow controllers to sample current traffic.

However energy monitoring for networks is challenging due to the dynamics of infrastructure information and special measurement mechanisms. Therefore, a complete energy-aware information model is important for OpenNaaS to describe all the measurement information and meta information for monitoring and to exchange the energy information between software components. Based on the analysis above, we implemented both energy monitoring capabilities and energy description capabilities:

Energy monitoring capabilities obtain and provide energy usage information from power meters for the observed metrics: (power, energy, CO2 emission rate, and electricity price) and from data statistics for the calculated metrics: (total Electricity, CO2 emission, and energy efficiency). The measurement data is saved in an OpenNaaS database, as shown in Fig. 5.2. The capabilities are responsible for communication with power meters. We have created power meter drivers for SNMP access to different power meter vendors e.g. Rackitivity and APC PDUs. Only numeric Object Identifiers (OIDs) in the drivers are different for different PDUs. Each OID identifies a variable that can be read or set via SNMP. The capabilities not only measure the power usage of a single device, but also monitor the power usage of a network route. Using SNMP, the capabilities can obtain and control the power state of switches and ports.

Energy description capabilities describe and create meta information about energy source, power meters, green metric, power state etc. in OpenNaaS. With EDL, OpenNaaS instantiates energy information using a common vocabulary and makes information understandable between software components. An important piece of information that EDL can describe is the relationship of power meters and remote network devices, so that it is understandable which measured energy usage information from the port of a power meter belongs to which remote device.

The third set of capabilities we developed is the ***Green routing capabilities***, needed to calculate the green routing path. Three green metric options are available for routing: power consumption, electricity cost and CO2 emission, as these metrics are provided by the monitoring capabilities. In our original implementation we adopted a greedy routing algorithm. Once an OpenNaaS user selects the green metric to optimize upon, the algorithm traverses all the possible network routes between the original host and destination host of the flow and it chooses the route with the lowest value. The calculated route is converted to a list of OpenFlow flows in the specific Json format according to OFCs, and then the OpenFlow bundle sends the list to the OFCs to create flow forwarding rules.

OpenNaaS includes a model for the description of OpenFlow route tables and network routes, which are used by green routing capabilities. The switch ports are identified by number, and a route table is defined by: *IP source*, *IP destination*, *Source switch identifier, also named Datapath identifier (DPID)*, *Input port of the switch* and *Output port of the switch*. The output port identifies which switch the traffic is sent to on the next hop. A route or a routing path is a list of route tables.

Similar to route tables in OpenNaaS, OpenFlow flows also include the identifier of switches and the output ports of switches. OpenNaaS has knowledge of the network topology that also depicts the connections between OFCs and switches. So even if there are multiple OFCs in a network, the OpenFlow bundle knows which switch the flow belongs to and which OFC controls this switch. OpenNaaS can add the flow rules to the correct OFC.

5.5.3 OpenFlow Controllers

In our design, after the route is calculated OFCs insert flow forwarding rules for the route in a proactive way. The *static routing module* pushes all the flow entries to the switches before traffic arrives, to save the time of processing routing requests from all the switches. We didn't change the static routing module and its REST APIs in the OFCs. The OpenFlow bundle in the OpenNaaS server calls different APIs according the type of an OFC: Static Flow Pusher APIs in Floodlight and Static Routing APIs in OpenDaylight.

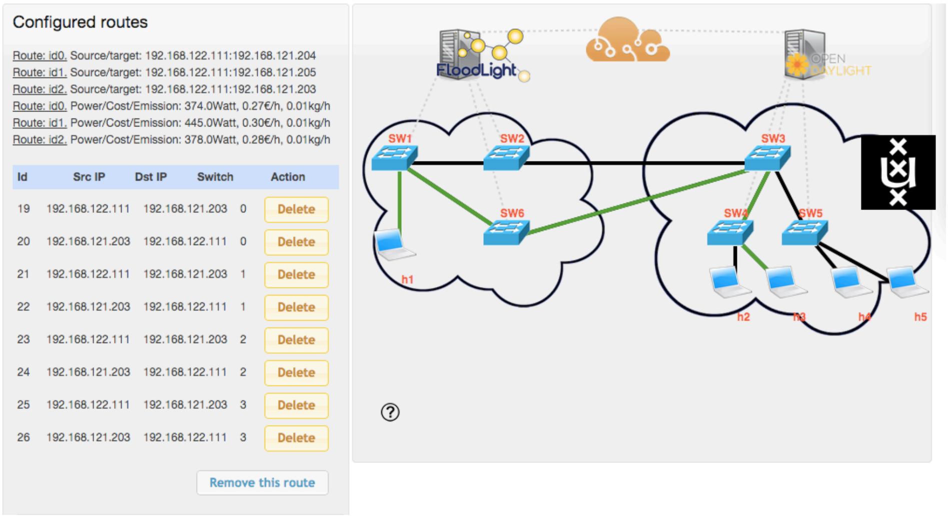
To support the second mode of invocation, we created the *event detection module* in the controllers to forward routing requests from switches to OpenNaaS.

The module detects a packet-in event, and then sends a REST routing request to the energy-aware bundle. The message contains the source and destination IP of the packet, the DPID of the switch and the input port where the packet enters the switch.

5.6 Prototype

We developed a prototype to show the functionality of energy-aware OpenNaaS. The prototype includes a web client GUI, which communicates with the capabilities of OpenNaaS through REST APIs.

We emulated a Mininet network with a topology of 6 OpenFlow switches and 5 hosts, shown in Fig. 5.3. The switches have the same network capacity and are divided into two groups, controlled by Floodlight and OpenDaylight controllers respectively. To simulate different CO2 emission rate [126], we assumed the two groups consume solar and thermal energy respectively. We ran the OpenNaaS server in the same machine.


Configured routes

Route: id0. Source/target: 192.168.122.111:192.168.121.204
Route: id1. Source/target: 192.168.122.111:192.168.121.205
Route: id2. Source/target: 192.168.122.111:192.168.121.203
Route: id0. Power/Cost/Emission: 374.0Watt, 0.27€/h, 0.01kg/h
Route: id1. Power/Cost/Emission: 445.0Watt, 0.30€/h, 0.01kg/h
Route: id2. Power/Cost/Emission: 378.0Watt, 0.28€/h, 0.01kg/h

Id	Src IP	Dst IP	Switch	Action
19	192.168.122.111	192.168.121.203	0	Delete
20	192.168.121.203	192.168.122.111	0	Delete
21	192.168.122.111	192.168.121.203	1	Delete
22	192.168.121.203	192.168.122.111	1	Delete
23	192.168.122.111	192.168.121.203	2	Delete
24	192.168.121.203	192.168.122.111	2	Delete
25	192.168.122.111	192.168.121.203	3	Delete
26	192.168.121.203	192.168.122.111	3	Delete

Figure 5.3: A screen shot of the topology and the configured route in the emulated network environment

Fig. 5.4 presents a flowchart of the usecase in our emulated environment. A provider creates a set of abstract OpenFlow resources and loads the OpenFlow capabilities and energy-aware capabilities for the resources using OpenNaaS (*Step 1*).

After creating the resources, the provider sets the energy source information for the network, as well as the connections between switches and the outlets of power meters (*Step 2*). For our prototype, we interfaced OpenNaaS with an actual power meter to provide the power readings of emulated resources. At this point the provider sends the monitoring request with the specific metric (*Step 3*). OpenNaaS translates this request and forwards an SNMP sampling command to the power meter (*Step 3.1*) and reads the measurement data out (*Step 3.2*). The

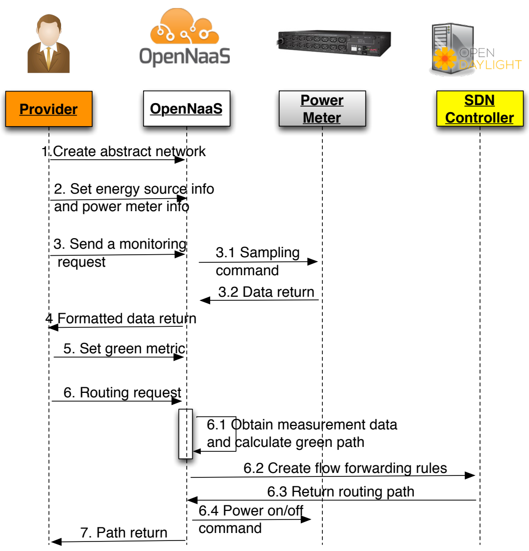


Figure 5.4: The flowchart of a general usecase of energy-aware OpenNaaS (Step 1 to Step 7)

data is described by EDL, and data is saved in a database or is sent back to the provider (*Step 4*). Table 5.1 shows the energy information for switch SW1 at a certain moment.

Table 5.1: An example of switch (SW1) information monitored

DPID	00:00:64:87:88:58:f6:57
Controller IP	controllersVM
Controller Port	8080
Power Consumption	70.0 Watt
Energy Source	
ID	Solar1
Price	0.002 Euros/kWh
CO2 emission rate	0.0015 kg/kWh

Our prototype also proves that OpenNaaS can measure the energy usage information of a specified routing path. Fig. 5.3 lists all the configured routes in the network and presents the total power consumption, electricity rate and emission rate of each route.

At this point the provider decides on a green optimization metric (*Step 5*) and submits a request to OpenNaaS for a path between a source and a destination (*Step 6*). OpenNaaS obtains the energy information of all the possible network routes between the end points; if the information is not available in the database, it will send a monitoring request immediately to obtain real-time energy information (*Step 6.1*). OpenNaaS selects the route with the least value and interacts with the OFCs to create flow forwarding rules in the switches (*Step 6.2*). At the end, the formatted route information is returned to the provider if the flow rules are successfully created (*Step 6.3*). According to the result route, OpenNaaS sends power on/off command to the power meter via SNMP and then the power meter changes the state of switches and links (*Step 6.4*). Fig. 5.3 shows a routing path we found between host1 and host3 using energy-aware OpenNaaS.

The benefits for adopting energy-aware OpenNaaS are many-fold. First, OpenNaaS has lightweight management costs. Network users and providers can configure their network through unified capabilities published in OpenNaaS that are transparent to specific technologies and hardware details. The control centralized in the OpenNaaS server manages the network as a whole rather than as a number of individual devices. Second, energy-aware OpenNaaS allows network users and providers to understand their energy usage information. They can profile the energy information for their network for further analysis, e.g. troubleshoot power shortage problems in data centers and prepare for the design a new network infrastructure. Network providers can even implement their own power management methods with the profiling information.

Our original prototype implemented a simple usecase where OpenNaaS creates a path with a greedy routing algorithm when receiving a routing request. A similar greedy algorithm was proposed in the previous study [78]. For the next version of energy-aware OpenNaaS we decided to further improve its energy awareness; for this we evaluated a set of routing algorithms and designed an optimized green routing strategy to replace the original greedy algorithm.

5.7 Energy-aware Routing Strategies

Energy-aware routing is a bin-packing problem, and it is not possible to find an optimal solution, especially for large scale networks. Therefore, rather than directly solving the energy-aware routing problem, we analyze the energy efficiency of common routing algorithms that already exist. We use the routing algorithms to find routes, and the switches and links on unused routes will be powered off.

Traffic aggregation in these routing algorithms could make flows run on the same link. A flow scheduling algorithm is needed to schedule flows. So we will also analyze the energy efficiency of flow scheduling algorithms as post of optimized energy-aware routing strategy.

5.7.1 Flow Routing

As discussed in Sec. 5.2, solutions to the power minimization problem rely on the assumption that the completion time of traffic is constant. They find a routing path for each flow and combine all the result paths to generate a subset of the network topology. They are power efficient but could effectively be energy inefficient. *Power-greedy routing* is a simple solution to the power minimization problem. For each flow, power-greedy routing evaluates all possible paths with sufficient capacity and selects a routing path with the least increase of total network power consumption.

Shortest path routing pursues low-delay and high throughput for data transmission. It selects a routing path with the least number of lengths or weights.

Shortest path routing is not power-aware, but it could lead to low completion time.

In data center networks, networks are over-provisioned. Multiple paths with similar length between two hosts are likely to exist, so their performance is similar, e.g. there are multiple parallel paths between any two hosts in a BCube network and some of paths have the same length [74]. The same applies to a Fat Tree network. If we observe Fig. 5.5, which is an example of partial Fat Tree network, we see that 4 shortest paths exist between any two hosts. For example, the shortest paths between *Host1* and *Host5* are as follows:

path1: SW7→ SW5→ SW1→ SW9→ SW11
path2: SW7→ SW5→ SW2→ SW9→ SW11
path3: SW7→ SW6→ SW3→ SW10→ SW11
path4: SW7→ SW6→ SW4→ SW10→ SW11

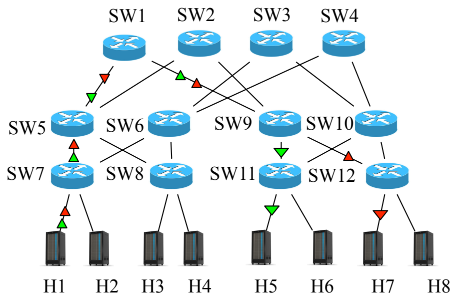


Figure 5.5: An example of partial Fat Tree network

It is reasonable to select an energy efficient path among multiple shortest paths. Thus, we propose *priority-based shortest routing*. The priority of a routing path is differentiated by the use of switches in the path. The path that includes the highest number of currently used switches has the highest priority. Priority-based shortest routing computes multiple shortest routing paths for a flow and selects the one with the highest priority. For example, if *SW6* and *SW4* are currently working or not being powered off, *path4* has the highest priority and should be selected. The basic reasoning is to use as few network resources as possible to achieve the same network performance. *Random routing*, which randomly selects a routing path from possible paths, is the baseline for the three other routing algorithms.

5.7.2 Flow Scheduling

Traffic aggregation in flow routing might make multiple flows run on the same link and share the overall bandwidth. Exclusive flow scheduling (EXR) is an alternative to link sharing; EXR transfers the flows one by one and only allows each flow exclusive use of the overall link bandwidth. We can prove that EXR is a more efficient scheduling algorithm comparing to link-shared scheduling. This is well illustrated with an example based on Fig. 5.5. Let's assume that two flows are transferred at the same time in this Fat Tree network, namely "flow1: H1 to H5" and "flow2: H1 to H7" with size of s_1 and s_2 respectively. For simplicity, we assume that only a set of switches (*SW7, SW5, SW1, SW9, SW11 and SW12*) and only the links with data transmission are currently active. The completion time of two flows are t_1 and t_2. Let α be the proportion of bandwidth used by *flow1* and $1-\alpha$ be the proportion used by *flow2*. We assume the flow1 is finished first (The result can be proved accordingly if the flow2 is finished first), so the range of α is as follows:

$$\frac{s_1}{s_1+s_2} < \alpha \le 1 \tag{5.6}$$

Since most data center networks employ a homogeneous set of switches to interconnect servers, we assume all the switches have same static power P_{base} and all the links of the switch have the same power model $p(u)$. The network energy E of transmitting flow1 and flow2 can be calculated as follows:

$$\begin{aligned}
E &= E_{SW7} + E_{SW5} + E_{SW1} + E_{SW9} + E_{SW11} + E_{SW12} \\
&= (P_{base} + p(\alpha C)) \cdot t_1 + 5P_{base} \cdot t_2 + 4p(C) \cdot t_2 \\
&+ p(C)) \cdot (t_2 - t_1) + p((1-\alpha) \cdot C) \cdot t_1 \\
&= (P_{base} + p(\alpha C)) \cdot \frac{s_1}{\alpha C} + 5P_{base} \cdot \frac{s_1+s_2}{C} \\
&+ 4p(C) \cdot \frac{s_1+s_2}{C} + p(C)) \cdot (\frac{s_1+s_2}{C} - \frac{s_1}{\alpha C}) \\
&+ p((1-\alpha) \cdot C) \cdot \frac{s_1}{\alpha C} \\
&= \frac{P_{base}}{C}(5s_1 + \frac{s_1}{\alpha C} + 5s_2) + \frac{p(\alpha C)}{\alpha C} s_1 \\
&+ \frac{p(C)}{c}(5s_1 - \frac{s_1}{\alpha} + 5s_2) + \frac{p((1-\alpha) \cdot C)}{\alpha C} s_1
\end{aligned} \tag{5.7}$$

In the real networking devices, the power consumed by the link is small compared to the static power consumption; the power consumption of an idle link is quite close to that of a full load. We can assume $p((1-\alpha) \cdot C)$, $p(\alpha C)$ and $p(C)$ are equal. Therefore, the Eq. 5.7 can be simplified as follows:

$$E = \frac{P_{base}}{C}(5s_1 + \frac{s_1}{\alpha} + 5s_2) + \frac{p(C)}{C}(5s_1 + \frac{s_1}{\alpha} + 5s_2) \tag{5.8}$$

When α equals to 1, Eq. 5.8 reaches a minimized value. So, the network has the least energy consumption when a flow is transferred exclusively on a link.

Based on this analysis, we combined priority routing and EXR scheduling to create a complete and optimized energy-aware routing strategy. In the next section we will validate our choice by conducting comparative studies by simulation.

5.8 Evaluation

In our simulation we use two common data center topologies – Fat Tree using 4-port switches (Fat Tree(4)) and 3-level BCube using 2-port switch (BCube(2,3)).

The capacity of each link in the simulated topologies is 1Gbps; the total number of server is 16 servers. There are 32 switches in BCube(2,3) and 20 switches in Fat Tree(4). We present two groups of flows with totally different flow size to study their potential in extreme situations. We simulate the flows with a typical size (64MB) in Hadoop data centers and a large size (5GB) in scientific data centers. In each group of flows, the size of each flow changes in a pretty small range (less than 1%). The source and destination servers of each flow are random. The arrival rate of flows follows the Poisson distribution. We simulate the flows coming in one minute and the maximum number of flows are about 3000. The average throughput of flows is 500MB/s.

All the switches in the same network are homogeneous. According to the power profiling of networking devices in our experiments [139] and from other studies in [78], we assume the dynamic power consumption accounts for less than 10% of total power consumption. We use a linear model to capture the relation between dynamic power consumption and network load.

For the simplicity of our simulation, we focus on small-sized networks. This is because power-greedy routing is not scalable, we have to calculate all possible paths (without loops) and compare their power consumption. The algorithm to find all the paths has a computational complexity of $O(V!)$ [37]. V represents the number of switches in the network. Priority-based shortest routing and shortest routing are both fundamentally scalable. Using the Dijkstra algorithm to find a shortest routing has a computational complexity of $O(V^2)$. Priority-based shortest routing finds n shortest routing paths, and n is a small integer usually. Its computational complexity is $O(n^3 \cdot V(E+V \log V))$, where E is the number of links in the network.

We explore how much energy to consume for data transmission with different routing strategies that include the combination of routing algorithms and flow scheduling algorithms. The flow scheduling algorithms are differentiated by whether flows are allowed to share link bandwidth. We define the *Energy Efficiency* metric as the amount of data transmission per unit of energy consumption. The *Mean Flow Completion Time* is the mean time interval between arrival and completion of flows, used to qualify network performance.

Fig. 5.6 (a) and (b) show network energy efficiency and mean flow completion time by the different routing algorithms against the big flow arrival rate in the BCube network. The solid lines represent the strategies with shared flow scheduling and the dotted lines describe the strategies with EXR. In Fig. 5.6 (a), we observe that the BCube network's energy efficiency rises with the growth of the arrival rate. It proves higher network utilization makes the network more energy efficient. The strategies with EXR offer significant energy saving compared to the strategies with link sharing. For example, priority-based routing with EXR is about 35% higher energy efficiency over priority-based routing with link sharing. In regard to the same scheduling algorithm, shortest routing has a growing advantage (at most 20%) on the energy efficiency than greedy routing when the flow arrival rate increases. The energy efficiency of priority-based routing is about 5% higher than that of shortest routing when the arrival rate of flows is over 10.

In Fig. 5.6 (b), we observe that the strategies with link sharing have longer mean flow completion time than the strategies with EXR. Priority-based and shortest routing algorithms have similar performance, which are better than greedy

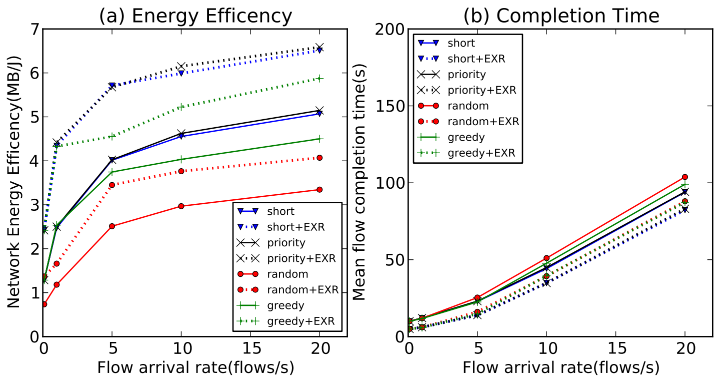


Figure 5.6: Network energy efficiency and mean flow completion time against the arrival rate of large flows(5GB) in the BCube network

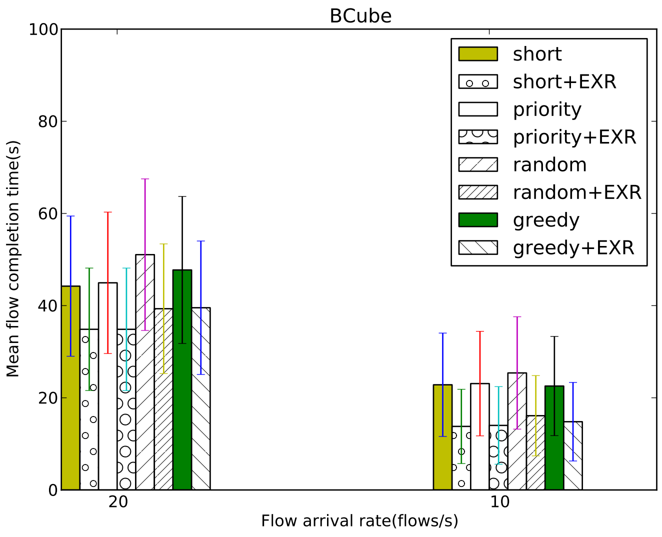


Figure 5.7: Mean flow completion time with standard error against the arrival rate of large flows(5GB) in the BCube network

routing. During exclusive scheduling, the performance of greedy routing is as bad as random routing.

Fig. 5.7 shows the standard error of completion time for all the flows. The completion time of the strategies with link sharing change in a similar range as the strategies with EXR.

Fig. 5.8 (a) illustrates that network energy efficiency can improve 2.5 times when the big flow arrival rate increases from 0.1 to 20 in the Fat-Tree network. The strategies with EXR still offer an obvious improvement of energy efficiency over the strategies with link sharing. Although priority-based shortest routing

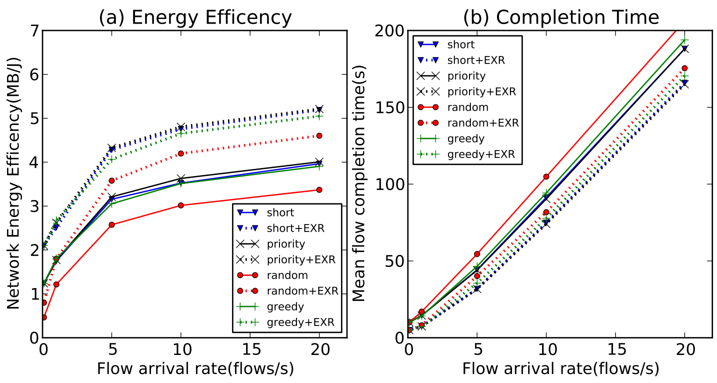


Figure 5.8: Network energy efficiency and mean flow completion time against the arrival rate of large flows(5GB) in the Fat Tree network

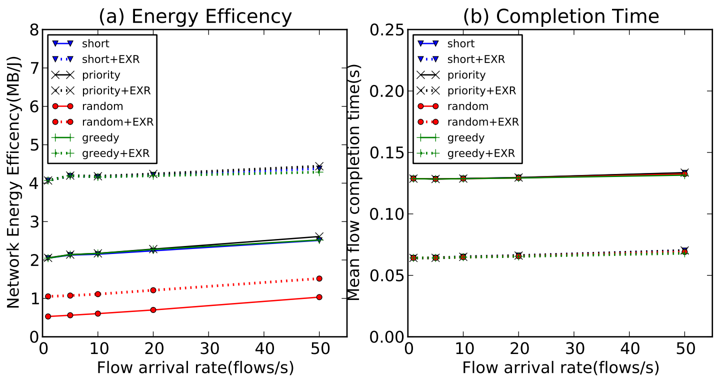


Figure 5.9: Network energy efficiency and mean flow completion time against the arrival rate of small flows(64MB) in the BCube network

is the most energy efficient most of the time, the advantage of the shortest and greedy routing algorithms in the Fat Tree network is slight compared to that of the BCube network.

Fig. 5.8 (b) shows the mean flow completion time against the big flow arrival rate in the Fat-Tree network. The strategies with link sharing are worse than the strategies with EXR. The mean flow completion time using the priority-based algorithm and using the shortest routing algorithm are nearly the same; they are smaller than that of the other two routing algorithms.

Fig. 5.9 (a) and (b) show network efficiency against the small flow arrival rate in the BCube network. The two figures reiterate that the strategies with link sharing are less energy efficient and have longer completion time than the strategies with EXR.

This is also observed in Fig. 5.10 (a) and (b) for the BCube network. Compared to the experiments using traffic with big flows, the energy efficiency of the BCube and Fat Tree networks improves less (at most 1 time) when the small flow

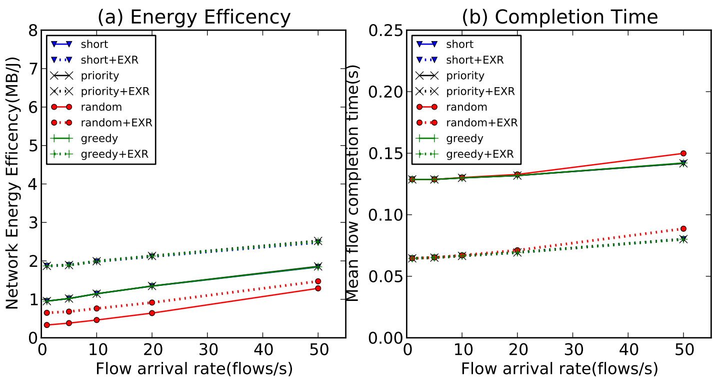


Figure 5.10: Network energy efficiency and mean flow completion time against the arrival rate of small flows(64MB) in the Fat Tree network

arrival rate increases; the energy efficiency of the networks for small flow transmission are lower. Energy efficiency of the shortest, priority-based and greedy routing algorithms are similar, and the mean flow completion time of these routing algorithms are the same. Although these routing algorithms could compute different routing paths for the same flow, the difference of route selection impacts the network power consumption quite lightly for small flows because they are completed in very short time.

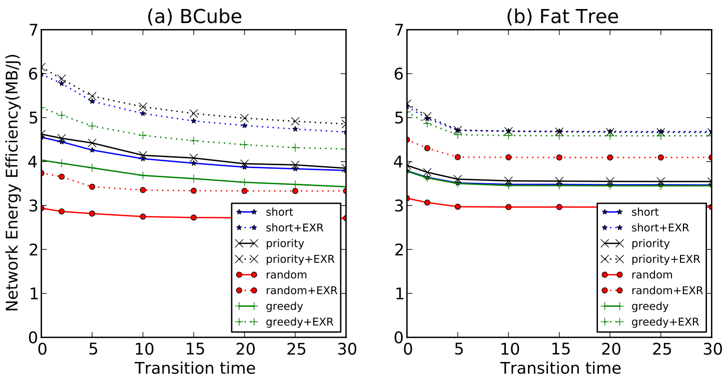


Figure 5.11: Network energy efficiency against transition time of switches under large flows(5GB) in Fat Tree and BCube networks

In the above experiments we assume no transition time for switches. But in a real network environment, the switches usually take a few seconds to power on/off after receiving off/on signals [107]. In order to simulate the vendors with different transition times, we study network energy efficiency against the transition time. A switch doesn't goes off if the arrival interval between two flows along it is less than the transition time. The flow arrival rate is fixed at 10 flows per second and the size of all the flows is 5GB. The result of Fat-Tree and BCube networks is shown in Fig. 5.11 (a) and (b). We can find that network energy efficiency declines slowly and remains stable when the transition time is large enough. In regard to the same scheduling algorithm, we can observe that the energy efficiency of priority-based routing is about 5% higher than that of shortest routing, while shortest routing has about 5%-20% higher energy efficiency than greedy routing.

We conclude all the experiments as follows. Power-greedy routing is a power-minimization algorithm, but it could not find an optimal routing for all the flows because its result is in essence a local optimization. Besides, power-greedy routing degrades the completion time of flows. That is the reason that our experiments indicate that the power-greedy algorithm is less energy efficient. Exclusive flow scheduling is much more energy efficient than scheduling with link sharing regardless of which routing algorithm is used. The strategy we highlight is the combination of priority-based shortest routing and exclusive flow scheduling. It has the obvious improvement of energy efficiency for big file transmissions in the BCube network. In this case, the priority-based routing with EXR is about 5%-35% higher energy efficiency than other common strategies. When the transition time of switches is considered, it is still more energy efficient, compared to either the routing algorithms or the scheduling algorithms used. The network performance of this strategy doesn't degrade and its mean flow completion time is nearly same as the strategies with shortest routing.

5.9 Discussion

We will implement the exclusive flow scheduling algorithm in the prototype in future. OpenNaaS and OFCs need to be modified to enable exclusive flow scheduling. OpenNaaS has to define two flow lists to monitor active flows and suspended flows. OpenNaaS first has to calculates the routing path and configure forwarding rules for any new incoming flow; then, it needs to determine the scheduling state of the flow according to the state of links in the path. If part of the links in the path are used, OpenNaaS must put the flow into the suspended list and asks the OFC to suspend the flow. If all links in the path are available, OpenNaaS can add the flow into the active list and notify the OFC to permit the flow. When an active flow is finished, the switch has to generate the completion event and forward it to OpenNaaS through the OFC. Finally, OpenNaaS can fetch a suspended flow from the suspended list, and allow flow transmission.

Some general features of EXR should also be carefully considered. Exclusive flow scheduling can improve on the mean flow completion time but could increase this value for small flows, compared to shared flow scheduling. Therefore, for the applications which have strict QoS e.g. response time for small flows, exclusive flow scheduling is not the best option even if it does save energy consumption. NERNs are the important users of OpenNaaS, and the majority of applications hosted by NERNs are scientific computing applications that are tolerant of execution time. So exclusive flow scheduling can meet their requirements. Also, the algorithm for fetching the suspended flow could be first-in-first-out, smallest or biggest flow first etc. We can choose one according to QoS requirements in future implementations. Large-sized flows easily lead to network congestion [96], so giving large flows higher priority can further improve the mean network performance, while giving small flows higher priority can reduce the delay in completing them.

Energy-aware routing strategies power off unused network components and they impact the oversubscription rate of date center networks. If a burst of traffic arrives, the unused network components require some time to become available. So in this case energy-aware routing strategies could increase the response time of data transmission. Enabling energy-aware routing strategies only at the non-peak time can lower this negative impact. During this period, the tolerance for link failure is also higher than at peak time.

5.10 Conclusion

OpenNaaS is a network management platform that includes SDN support and interworks several SDN platforms to orchestrate network services; this makes it a suitable management platform for adoption in data centers moving to SDN. The energy-aware OpenNaaS we developed for energy-aware routing builds on the consolidated OpenNaaS framework, and it's the first implementation of this kind as far as we know. Our prototype shows we can also measure the energy, cost and sustainability information of networks for providers or users.

In the prototype, we discuss the improvement of routing strategies by combining flow routing and flow scheduling. Priority-based shortest routing finds the most power efficient routes from multiple shortest paths. Exclusive scheduling speeds up all flows on the same link. The simulation results show that combined strategy can effectively improve network energy efficiency, in particular when traffic consists of large-sized flows.

Chapter 6

Conclusion

e-Infrastructures are ICT infrastructures hosting networks, data centers and collaborative environments, developed to support scientific research in a distributed research community. To meet the requirements of processing big data, the size and scale of data centers and networks are increasing. This leads to large volumes of energy consumption and GHG emissions, which motivated us to study and improve upon energy monitoring and energy management suitable for e-Infrastructures.

The distributed environment in an e-Infrastructure requires the exchange of knowledge to map applications onto the infrastructure. The first challenge we addressed was creating a semantic information model that describes energy-related knowledge. Then we presented a proper approach for building an information system for organizing energy knowledge based on the model. Leveraging our semantic information models and information system, ultimately we validated new algorithms and software implementations for energy management. We focused on power consumption estimation in servers and energy-aware routing for networks.

6.1 Semantic Information Model

In Chapter 2 and Chapter 3, we answered the first research question – *Q1: What is the proper approach to design and create an energy-aware information model for the description of e-Infrastructures and develop a sufficient information system for their energy monitoring?* Chapter 2 presented how we built the energy-aware semantic model – the Energy Description Language. Chapter 3 described how we developed the energy-aware information system – the Energy Knowledge Base.

In Chapter 2 we introduced the RDF model in the Semantic Web and presented the benefits of using an semantic approach to describe e-Infrastructures with energy awareness. RDF improves data interoperability in a distributed environment and OWL ontologies are extensible.

Then we described our group's previous work on the semantic model for computing and network infrastructures – the Infrastructure and Network Description Language. INDL is defined as an extension of Network Markup Language (NML), thus creating a extensible, technology independent model of computing infrastructures. INDL has been a basis for modeling many different distributed infrastructures: the CineGrid infrastructure, the NOVI federated platforms and the GEYSERS architecture. It was also the basis for ExoGENI. In essence, INDL is a suitable model for e-Infrastructures. The use of Semantic Web technology in the INDL ontology facilitates the creation of models that can be easily connected, stacked and extended by other models.

After that, we examined the state of the art on energy-aware information models for infrastructures, and found that none of these representations satisfy our requirements for energy management in distributed environments. Thus, we presented our approach of building EDL upon INDL. EDL itself focuses on the concepts and relationships of energy monitoring and measurement. It contains the *Green Metric* part that describes measurement data in different energy metrics, the *Characteristic* part that describes non-measurable state of computing or networking resources for green resource discovery and the *Monitor Component* part that describes the way to obtain measurement data of resources from sensors and the way to organize measurement data in logs. The EDL ontology is also an OWL-based model. The concepts in EDL can support a wide range of power management scenarios such as power estimation and green resource discovery.

In Chapter 3 we developed the Energy Knowledge Base system for energy monitoring to manage and organize energy-aware data and metadata in e-Infrastructures. As far as we know, this is the first attempt to implement a semantic information system with energy awareness in a distributed infrastructure. An EKB agent works in the local domain. EKB can work in a distributed environment as it contains multiple agents connected by the Aggregation Service. The EKB agent is composed of knowledge components, which represent knowledge from the infrastructure and from the various information sources, and software components, which are responsible for collecting and providing knowledge. EKB leverages EDL to instantiate semantic description, which makes data interoperable. In the process of instantiation, we validated the concepts in the Energy Description Language ontology. We also presented the role of EKB in two energy management usecases: the Power Budget Calculator and green resource discovery. We evaluated the scalability of two EKB implementations when the number of users increases from 1 to 100. Our experiments show that the EKB implementation which stores measure data in a time-series data base instead of a triple store, is more scalable.

6.2 Energy Management

With the help of the EDL ontology and EKB information system, we have developed our specific research on energy management in e-Infrastructures. In Chapter 4 and Chapter 5, we answered the second research question – *Q2: What new energy management techniques will emerge by applying the developed information model and information system?*

In Chapter 4 we answered subquestion – *Q2a: How do we build and evaluate non-linear approaches for power estimation in a cluster environment?* We applied neural network models and unsupervised classification models with basic OS-reported resource features for power estimation. We trained and evaluated the power estimation models in a cluster environment using a large amount of measurement data from EKB; and we evaluated power estimation models in terms of not only accuracy but also portability and usability. The important results we got are as follows.

1. The GMM approach has the best power estimation accuracy compared with the neural network approaches and the linear approaches in an I/O and CPU-intensive mixed environment; GMM is not suitable in a purely CPU-intensive environment, but the optimized GMM approach, which manually clusters each data set into 5 groups according to the CPU usage, can achieve similar performance compared to the two neural networks.
2. Compared to GMM, the neural network approaches have bad portability when they are applied to homogeneous nodes across the same or other clusters.
3. GMM needs the longest training time. We suggest to use GMM or optimized GMM if one focuses on the estimation accuracy, and to use a multiple variable linear regression in time constrained environments.

In Chapter 5 we answered the subquestion – *Q2b: What is a proper way to explore energy-aware routing in data center networks and how much impact do routing strategies have on the energy efficiency of the networks?* We presented a prototype of energy-aware OpenNaaS, which is a solution to energy-aware routing in a network management platform for the purpose of improving network energy efficiency. Energy-aware OpenNaaS inherits and enhances powerful network management capabilities, e.g. dynamically obtaining power and topology information, and supporting multiple types of SDN controller, so it has wide applicability. Energy-aware OpenNaaS leverages EDL to define data elements and their structure for monitoring capabilities. It can measure the energy, cost and sustainability of networks for providers or users. In the prototype, we studied the effect of routing strategies that combines flow routing and flow scheduling on network energy efficiency and performance. Priority-based shortest routing finds the most power efficient routes from a set of multiple shortest paths. Exclusive scheduling speeds up the flows on the same link. The simulation results show that the combination of priority-based shortest routing and exclusive scheduling can effectively improve energy efficiency (5%-35%) of the BCube network without performance degradation when traffic consists of large-sized (5GB) flows.

6.3 Road Ahead

We see three directions in which our work can be extended: creation of information models for access policies, support for automatic information integration in the Energy Knowledge Base, and improvements of power estimation approaches.

The focus of this thesis is mainly on infrastructure models and monitoring models. INDL is a model to describe the structure and configuration of infrastructures and EDL is an attempt to model the energy-related states of the infrastructure. Besides a infrastructure model and a monitoring model, a complete resource management system also needs an access policy model to describe which types of actions are allowed on which resources and which actors are allowed to perform these actions [122].

The energy management technique we have discussed in Chapter 5 relies on consolidating networking devices and components by local operators, which is a simple scheduling scenario. An energy management system can easily carry out this activity on networks without an access model. But when considering complicated energy management scenarios in future, e.g. where operators can manage remote infrastructures by combining frequency scaling and consolidation, the scheduling activities are not straightforward. In this case, it is essential to investigate an access policy model for energy management based on our work.

EKB can support fixed information sources for obtaining measurement and metadata of infrastructures. In our vision, one of the research directions for EKB is to allow automatic information integration. EKB automatically creates and updates the description of information sources e.g. addresses and APIs using ontologies, and the description dynamically matches with the registered information sources. To enable this, information source ontologies should be modified to include the mapping of the models of information sources to EDL such that EKB can know how to instantiate native data from those sources. A new agent that manages the information sources is also needed.

Although the DAS-4 cluster is not a production cluster, our study on designing and evaluating power estimation approaches in Chapter 5 can be useful for future work in production clusters. A point for future research is to study the impact of different subsets of resource features and time series on the approaches for measuring the accuracy of power estimation. In theory, two factors could influence the power estimation.

1. Using a feature, e.g. memory utilization, in the model or not is totally distinct in theory. It is worth knowing how much accuracy will degrade if we don't maintain data about a feature.
2. Workloads can be characterized not only by resource features but also sample time; a workload could show a close relation with time, e.g. Map is always followed by Reduce in a Hadoop cluster.

Another point for future research is to apply our approaches in the study of estimating GPU power consumption. Some features about GPUs, e.g. GPU memory usage, should be taken into consideration in order to precisely estimate the power consumption of GPU workloads [119].

Appendix A

OWL Schema of The Energy Description Language

This chapter describes the normative schema of the OWL syntax using the OWL ontology definition below.

```

<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY nml "http://schemas.ogf.org/nml/2013/05/base#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY edl "http://www.science.uva.nl/research/sne/edl#" >
    <!ENTITY indl "http://www.science.uva.nl/research/sne/indl#" >
]>


<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:edl="http://www.science.uva.nl/research/sne/edl#"
     xmlns:indl="http://www.science.uva.nl/research/sne/indl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:nml="http://schemas.ogf.org/nml/2013/05/base#">
    <Ontology rdf:about="http://www.science.uva.nl/research/sne/edl">
        <imports rdf:resource="file:indl.owl"/>
    </Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////
    //
    // Annotation properties
    //
    ///////////////////////////////////////////////////////////////////////
     -->




    <!-- http://schemas.ogf.org/nml/2013/05/base#parameter -->

    <AnnotationProperty rdf:about="&nml;parameter">
        <rdfs:domain rdf:resource="&nml;Node"/>
    </AnnotationProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.science.uva.nl/research/sne/edl#atPowerState -->

    <ObjectProperty rdf:about="&edl;atPowerState">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;PowerState"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#attachTo -->

    <ObjectProperty rdf:about="&edl;attachTo">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;Outlet"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasCapability -->

    <ObjectProperty rdf:about="&edl;hasCapability">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;PowerCapability"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasDriver -->

    <ObjectProperty rdf:about="&edl;hasDriver">
        <rdfs:range rdf:resource="&edl;Driver"/>
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasLoad -->

    <ObjectProperty rdf:about="&edl;hasLoad">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;Load"/>
        <rdfs:domain rdf:resource="&edl;MonitorLog"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasLog -->

    <ObjectProperty rdf:about="&edl;hasLog">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;MonitorLog"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasMeasurement -->

    <ObjectProperty rdf:about="&edl;hasMeasurement">
        <rdfs:domain rdf:resource="&edl;Load"/>
        <rdfs:range rdf:resource="&edl;Measurement"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasOutlet -->

    <ObjectProperty rdf:about="&edl;hasOutlet">
        <rdfs:range rdf:resource="&edl;Outlet"/>
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#hasUnit -->

    <ObjectProperty rdf:about="&edl;hasUnit">
        <rdfs:domain rdf:resource="&edl;Metric"/>
        <rdfs:range rdf:resource="&edl;Unit"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#informOf -->

    <ObjectProperty rdf:about="&edl;informOf">
        <rdfs:range rdf:resource="&edl;ObservedGMetric"/>
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#monitor -->

    <ObjectProperty rdf:about="&edl;monitor">
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
        <inverseOf rdf:resource="&edl;monitoredBy"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#monitoredBy -->

    <ObjectProperty rdf:about="&edl;monitoredBy">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;PowerMeter"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#useEnergySource -->

    <ObjectProperty rdf:about="&edl;useEnergySource">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&edl;EnergySource"/>
    </ObjectProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#useMetric -->

    <ObjectProperty rdf:about="&edl;useMetric">
        <rdfs:domain rdf:resource="&edl;Load"/>
        <rdfs:range rdf:resource="&edl;Metric"/>
    </ObjectProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////
     -->



    <!-- http://www.science.uva.nl/research/sne/edl#UDPaddress -->

    <DatatypeProperty rdf:about="&edl;UDPaddress">
        <rdfs:domain rdf:resource="&edl;Driver"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#capability -->

    <DatatypeProperty rdf:about="&edl;capability">
        <rdfs:domain rdf:resource="&edl;PowerCapability"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#electricityPrice -->

    <DatatypeProperty rdf:about="&edl;electricityPrice">
        <rdfs:domain rdf:resource="&edl;EnergySource"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#
                emissionPerUnitofEnergy -->

    <DatatypeProperty rdf:about="&edl;emissionPerUnitofEnergy">
        <rdfs:domain rdf:resource="&edl;EnergySource"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#endEpochTime -->

    <DatatypeProperty rdf:about="&edl;endEpochTime">
        <rdfs:domain rdf:resource="&edl;MonitorLog"/>
        <rdfs:range rdf:resource="&xsd;long"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#metricName -->

    <DatatypeProperty rdf:about="&edl;metricName">
        <rdfs:domain rdf:resource="&edl;Metric"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#metricValue -->

    <DatatypeProperty rdf:about="&edl;metricValue">
        <rdfs:domain rdf:resource="&edl;Measurement"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#model -->

    <DatatypeProperty rdf:about="&edl;model">
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#moduleId -->

    <DatatypeProperty rdf:about="&edl;moduleId">
        <rdfs:domain rdf:resource="&edl;Outlet"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#nodeName -->

    <DatatypeProperty rdf:about="&edl;nodeName">
        <rdfs:domain rdf:resource="&nml;Node"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#numberOfModule -->

    <DatatypeProperty rdf:about="&edl;numberOfModule">
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#numberOfOutlet -->

    <DatatypeProperty rdf:about="&edl;numberOfOutlet">
        <rdfs:domain rdf:resource="&edl;PowerMeter"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#outletId -->

    <DatatypeProperty rdf:about="&edl;outletId">
        <rdfs:domain rdf:resource="&edl;Outlet"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#ratedPower -->

    <DatatypeProperty rdf:about="&edl;ratedPower">
        <rdfs:domain rdf:resource="&edl;PowerState"/>
        <rdfs:range rdf:resource="&xsd;float"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#sampleDuration -->

    <DatatypeProperty rdf:about="&edl;sampleDuration">
        <rdfs:domain rdf:resource="&edl;MonitorLog"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#sampleInterval -->

    <DatatypeProperty rdf:about="&edl;sampleInterval">
        <rdfs:domain rdf:resource="&edl;MonitorLog"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#startEpochTime -->

    <DatatypeProperty rdf:about="&edl;startEpochTime">
        <rdfs:domain rdf:resource="&edl;MonitorLog"/>
        <rdfs:range rdf:resource="&xsd;long"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#state -->

    <DatatypeProperty rdf:about="&edl;state">
        <rdfs:domain rdf:resource="&edl;PowerState"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#timestamp -->

    <DatatypeProperty rdf:about="&edl;timestamp">
        <rdfs:domain rdf:resource="&edl;Measurement"/>
        <rdfs:range rdf:resource="&xsd;long"/>
    </DatatypeProperty>



    <!-- http://www.science.uva.nl/research/sne/edl#unitName -->

    <DatatypeProperty rdf:about="&edl;unitName">
        <rdfs:domain rdf:resource="&edl;Unit"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>



    <!--
    ///////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////
     -->




    <!-- http://schemas.ogf.org/nml/2013/05/base#Node -->

    <Class rdf:about="&nml;Node"/>



    <!-- http://www.science.uva.nl/research/sne/edl#BrownEnergy -->

    <Class rdf:about="&edl;BrownEnergy">
        <rdfs:subClassOf rdf:resource="&edl;EnergySource"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#CalculatedGMetric -->

    <Class rdf:about="&edl;CalculatedGMetric">
        <rdfs:subClassOf rdf:resource="&edl;GreenMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Driver -->

    <Class rdf:about="&edl;Driver">
        <rdfs:subClassOf rdf:resource="&edl;MonitorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Efficiency -->

    <Class rdf:about="&edl;Efficiency">
        <rdfs:subClassOf rdf:resource="&edl;CalculatedGMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#EnergyConsumption -->

    <Class rdf:about="&edl;EnergyConsumption">
        <rdfs:subClassOf rdf:resource="&edl;ObservedGMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#EnergySource -->

    <Class rdf:about="&edl;EnergySource">
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#GreenEnergy -->

    <Class rdf:about="&edl;GreenEnergy">
        <rdfs:subClassOf rdf:resource="&edl;EnergySource"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#GreenMetric -->

    <Class rdf:about="&edl;GreenMetric">
        <rdfs:subClassOf rdf:resource="&edl;Metric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#
                HardwareSensorComponent -->

    <Class rdf:about="&edl;HardwareSensorComponent">
```

```
        <rdfs:subClassOf rdf:resource="&edl;Sensor"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Load -->

    <Class rdf:about="&edl;Load">
        <rdfs:subClassOf rdf:resource="&edl;MonitorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Measurement -->

    <Class rdf:about="&edl;Measurement">
        <rdfs:subClassOf rdf:resource="&edl;MonitorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Metric -->

    <Class rdf:about="&edl;Metric">
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#MonitorComponent -->

    <Class rdf:about="&edl;MonitorComponent">
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#MonitorLog -->

    <Class rdf:about="&edl;MonitorLog">
        <rdfs:subClassOf rdf:resource="&edl;MonitorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#ObservedGMetric -->

    <Class rdf:about="&edl;ObservedGMetric">
        <rdfs:subClassOf rdf:resource="&edl;GreenMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Outlet -->

    <Class rdf:about="&edl;Outlet">
        <rdfs:subClassOf rdf:resource="&edl;HardwareSensorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PerfMetric -->

    <Class rdf:about="&edl;PerfMetric">
        <rdfs:subClassOf rdf:resource="&edl;Metric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PowerCapability -->

    <Class rdf:about="&edl;PowerCapability">
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PowerConsumption -->

    <Class rdf:about="&edl;PowerConsumption">
        <rdfs:subClassOf rdf:resource="&edl;ObservedGMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PowerFactor -->

    <Class rdf:about="&edl;PowerFactor">
        <rdfs:subClassOf rdf:resource="&edl;ObservedGMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PowerMeter -->

    <Class rdf:about="&edl;PowerMeter">
        <rdfs:subClassOf rdf:resource="&edl;HardwareSensorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#PowerState -->

    <Class rdf:about="&edl;PowerState">
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Quantity -->

    <Class rdf:about="&edl;Quantity">
        <rdfs:subClassOf rdf:resource="&edl;CalculatedGMetric"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Sensor -->

    <Class rdf:about="&edl;Sensor">
        <rdfs:subClassOf rdf:resource="&edl;MonitorComponent"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#
                SoftwareSensorComponent -->

    <Class rdf:about="&edl;SoftwareSensorComponent">
        <rdfs:subClassOf rdf:resource="&edl;Sensor"/>
    </Class>



    <!-- http://www.science.uva.nl/research/sne/edl#Unit -->

    <Class rdf:about="&edl;Unit">
    </Class>



    <!--
    ///////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////
     -->




    <!-- http://www.science.uva.nl/research/sne/edl#EmissionEfficiency -->

    <NamedIndividual rdf:about="&edl;EmissionEfficiency">
        <rdf:type rdf:resource="&edl;Efficiency"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#EnergyEfficiency -->

    <NamedIndividual rdf:about="&edl;EnergyEfficiency">
        <rdf:type rdf:resource="&edl;Efficiency"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#Nuclear -->

    <NamedIndividual rdf:about="&edl;Nuclear">
        <rdf:type rdf:resource="&edl;GreenEnergy"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#Solar -->

    <NamedIndividual rdf:about="&edl;Solar">
        <rdf:type rdf:resource="&edl;GreenEnergy"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#Thermal -->

    <NamedIndividual rdf:about="&edl;Thermal">
        <rdf:type rdf:resource="&edl;BrownEnergy"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#TotalElectricityCost -->

    <NamedIndividual rdf:about="&edl;TotalElectricityCost">
        <rdf:type rdf:resource="&edl;Quantity"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#TotalEmission -->

    <NamedIndividual rdf:about="&edl;TotalEmission">
        <rdf:type rdf:resource="&edl;Quantity"/>
    </NamedIndividual>



    <!-- http://www.science.uva.nl/research/sne/edl#Wind -->

    <NamedIndividual rdf:about="&edl;Wind">
        <rdf:type rdf:resource="&edl;GreenEnergy"/>
    </NamedIndividual>
</rdf:RDF>
```

Appendix B

List of Abbreviations

ANN Artificial Neural Network
BDII Berkeley Data Base Information Index
BoD Bandwidth on Demand
CIM Common Information Model
CPU Central Processing Unit
CUE Carbon Usage Effectiveness
DAS-4 Distributed ASCI Supercomputer 4
DCN Data Center Network
DMTF Distributed Management Task Force
DPID DataPath Identifier
DVFS Dynamic Voltage and Frequency Scaling
ECMP Equal-Cost Multi-Path
EDL Energy Description Language
EEE Energy Efficient Ethernet
EKB Energy Knowledge Base
EMAN Energy Management
EXR Exclusive Routing
GHG GreenHouse Gas
GIM Green Information Model
GLIF Global Lambda Integrated Facility
GMDH Group Method of Data Handling
GMM Gaussian Mixture Model
GPU Graph Processing Unit
GRE Generic Routing Encapsulation
GRNET Greek Research and Technology Network
GSN GreenStar Network
GSNONT GreenStar Network Ontology
GUI Graphical User Interface
IEEE Institute of Electrical and Electronics Engineer
IETF Internet Engineering Task Force
INDL Infrastructure and Network Description Language
IP Internet Protocol
MDS Monitoring and Discovery System
NIC Network Interface Card

NML Network Markup Language
NM-WG Network Measurements Working Group
NREN National Research and Education Network
OFC OpenFlow Controller
OID Object Identifier
OS Operating System
OSPF Open Shortest Path First
OWL Web Ontology Language
PBC Power Budget Calculator
PDU Power Distribution Unit
PM Physical Machine
PMC Performance Monitoring Counter
PUE Power Usage Effectiveness
QMA Query Manager Agent
RCDA Resource Collector and Description Agent
RDF Resource Description Framework
RDFS Resource Description Framework Schema
RMSE Root Mean Square Error
SDN Software Defined Network
SLA Service Level Agreement
SNMP Simple Network Management Protocol
SPARQL SPARQL Protocol and Query Language for RDF
SSD Solid State Disk
S-MDS Semantic Monitoring and Discovery System
S-OGSA Semantic Open Grid Service Architecture
TCO Total Cost of Ownership
TCP Transmission Control Protocol
TeS Test Set
TrS Training Set
TSDB Time Series Data Base
UML Unified Modeling Language
URI Uniform Resource Identifier
URL Uniform Resource Locator
vCPE Virtual Customer Premises Equipment
VM Virtual Machine
vLAN Virtual Local Area Network
VS Validation Set
XML Extensible Markup Language

Appendix C

Source Code Repositories

- An OWL file and Java library of the Infrastructure and Network Description Language:
  `https://bitbucket.org/uva-sne/indl`
- An OWL file and Java library of the Energy Description Language:
  `https://bitbucket.org/uva-sne/edl`
- Source codes of The Energy Knowledge Base, data and source codes of power estimation:
  `https://bitbucket.org/uva-sne/ekb`
- Source codes of the Energy-aware OpenNaaS, Source codes and a video of Green Routing Demo:
  `https://bitbucket.org/uva-sne/green-routing-demo`
- Source codes of the Green Routing Simulator:
  `https://bitbucket.org/uva-sne/green-routing-simulator`

List of Figures

1.1 GÉANT European connections [19] 5
1.2 The components of energy management in infrastructures. 7
1.3 Server power usage and energy efficiency at varying utilization levels, from idle to peak performance [44]. 8
1.4 (a) Costs distribution in a data center [72]; (b) Server peak power by hardware components from a Google data center [79]. 9

2.1 A simplified architecture of the Semantic Web 16
2.2 A simple example of RDF graph 17
2.3 Network Markup Language main classes and properties 19
2.4 INDL: Modeling internal node components. 20
2.5 INDL: Modeling virtualization of nodes. 20
2.6 Energy Description Language imports INDL – INDL and three main parts in EDL. They are the *Green Metric* classes and their properties, the energy*Characteristic* classes and properties and the *Monitor Component* classes and their properties 23
2.7 UML representation of Green Metric part 23
2.8 UML representation of energy-aware Characteristic part 25
2.9 UML representation of Monitor Component part 26

3.1 (a) Overview of the Energy Knowledge Base architecture (b) knowledge components and software components of an EKB Agent 34
3.2 Power consumption against TCP flow throughput for Huawei S1728GWR-4P [5]. 39
3.3 Total switch(es) power consumed for task (250Mbit) 40
3.4 An example of measurement data obtained from EKB for a node in DAS-4: (a) power consumption, (b) CPU utilization, (c) memory utilization, (d) disk IO time and (e) network speed 41
3.5 Processing time against the number of clients on two EKB implementations – EKB-TS and EKB-TSDB. 43
3.6 Response time against the number of clients on two EKB implementations – EKB-TS and EKB-TSDB. 43

4.1 The CDF distribution of CPU, memory and disk load in our cluster: nodes with GPUs vs. Hadoop nodes 51
4.2 Changes in average CPU, memory and disk load between consecutive 2-hour sampling periods: nodes with GPUs vs. Hadoop nodes vs. Google 51

4.3 An example of Group Method of Data Handling network structure . . . 53
4.4 The CDF distribution of RMSE of all the nodes in the test set 56
4.5 CDF of success ratios of all the nodes in ranges of the Δ 1% (a), 5% (b) and 10% (c) in the original test set 57
4.6 Mean RMSE of nodes for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort. (b) The Linpack test and Linpack-GPU test are both with a process grid ratio of 24. 58
4.7 Mean success ratio of nodes for the benchmarks. (a)(b) The MapReduce benchmark includes Writer and Sort. (c)(d) The Linpack test and Linpack-GPU test are both with a process grid ratio of 24. 59
4.8 Mean RMSE of nodes with GPUs for Linpack. The Linpack tests consume CPU usage in ranges of 25%, 50%, 75% and 100%. 60
4.9 The best and worst fit ratio using model migration for the benchmarks. (a) The MapReduce benchmarks includes Writer and Sort; (b) Linpack is tested with a process grid ratio of 24 61

5.1 OpenNaaS abstraction view: resources and capabilities. 69
5.2 Energy-aware OpenNaaS architecture 70
5.3 A screen shot of the topology and the configured route in the emulated network environment 72
5.4 The flowchart of a general usecase of energy-aware OpenNaaS (Step 1 to Step 7) 73
5.5 An example of partial Fat Tree network 75
5.6 Network energy efficiency and mean flow completion time against the arrival rate of large flows(5GB) in the BCube network 78
5.7 Mean flow completion time with standard error against the arrival rate of large flows(5GB) in the BCube network 78
5.8 Network energy efficiency and mean flow completion time against the arrival rate of large flows(5GB) in the Fat Tree network 79
5.9 Network energy efficiency and mean flow completion time against the arrival rate of small flows(64MB) in the BCube network 79
5.10 Network energy efficiency and mean flow completion time against the arrival rate of small flows(64MB) in the Fat Tree network 80
5.11 Network energy efficiency against transition time of switches under large flows(5GB) in Fat Tree and BCube networks 80

List of Tables

2.1 Comparison of existing energy-aware information models in the description of infrastructures. . . . 22
2.2 The predefined green metrics and their units in the EDL ontology . . . 24
2.3 Features of Energy Description Language in the description of infrastructures. . . . 27

3.1 Features of Energy Knowledge Base and common information systems. 32
3.2 Part of the REST APIs in the Energy Knowledge Base . . . 35
3.3 Parameters used for the task-based estimation tool . . . 40
3.4 A SPARQL query example: find the type of energy source a node is using . . . 41

4.1 Correlation coefficient of the resource features and power consumption obtained by measurement data of one typical node in the DAS-4 cluster. Part of feature types include 2 features due to two disks and two Network interface cards (NICs) equipped in one node. . . . 46
4.2 Summary of power estimation models for VMs and PMs; The default *accuracy* is on average if not specified . . . 49
4.3 The configuration of target nodes (two groups) in our cluster. . . . 50
4.4 Input parameters in the approaches and their optimal value for typical nodes with GPUs and Hadoop nodes . . . 55
4.5 Benchmarks and target resources for evaluation . . . 58
4.6 The training and estimation time cost, resource usage and TrS demand of the approaches . . . 61

5.1 An example of switch (SW1) information monitored . . . 73

General bibliography

[1] Distributed Management Task Force (DMTF). http://www.dmtf.org/.

[2] European Grid Infrastructure (EGI). http://www.egi.eu/.

[3] GÉANT. http://www.geant.org/.

[4] Global Lambda Integrated Facility (GLIF). http://www.glif.is/.

[5] Huawei S1728GWR-4P Switch. http://market.huawei.com/hwgg/enterprise/u-channel/pdf/S1700.pdf.

[6] Internet Engineering Task Force (IETF). http://www.ietf.org/.

[7] RDF Schema. http://www.w3.org/TR/rdf-schema/.

[8] The Distributed ASCI Supercomputer 4. http://www.cs.vu.nl/das4/home.shtml.

[9] The Semantic Web. http://www.w3.org/2001/sw/.

[10] EGEE: Enabling Grids for e-science. http://eu-egee-org.web.cern.ch/eu-egee-org/index.html, 2010.

[11] *IEEE Std 802.3az-2010 (Amendment to IEEE Std 802.3-2008)*, 2010.

[12] Google cluster trace, 2011.

[13] Notation3 (N3): A readable RDF syntax. http://www.w3.org/TeamSubmission/n3/, 2011.

[14] RDF 1.1 Turtle: Terse RDF Triple Language. http://www.w3.org/TR/turtle/, 2011.

[15] DMTF Common Informantion Model (CIM). http://dmtf.org/standards/cim, 2013.

[16] IETF: Energy Management (EMAN) model. https://datatracker.ietf.org/wg/eman/, 2013.

[17] Mantychore Project. http://www.mantychore.eu/, 2013.

[18] NOVI – Networking innovations over virtualized infrastructures. http://www.fp7-novi.eu/, 2013.

[19] e-irg white paper - best practices for the use of e-infrastructures by large-scale research infrastructures, 2014.

[20] ExoGENI. http://www.exogeni.net, 2014.

[21] Floodlight. http://www.projectfloodlight.org/, 2014.

[22] Google Data Centers. http://www.google.com/about/datacenters/efficiency/internal/, 2014.

[23] Nauge network. http://www.nuagenetworks.net/, 2014.

[24] NOX and POX OpenFlow controller. http://www.noxrepo.org/, 2014.

[25] Open platform for networks as a service. http://www.opennaas.org, 2014.

[26] OpenDaylight. http://www.opendaylight.org/, 2014.

[27] Ryu SDN framework. http://osrg.github.io/ryu/, 2014.

[28] Tail-f network control system. http://www.tail-f.com/network-control-system/, 2014.

[29] The Facebook Data Center FAQ. http://www.datacenterknowledge.com/the-facebook-data-center-faq/, 2014.

[30] The Green500. www.green500.org/, 2014.

[31] U.S. annual non-baseload CO2 output emission rate, year 2010 data. http://www.epa.gov/cleanenergy/energy-resources/refs.html, 2014.

[32] XML Schema 1.1. http://www.w3.org/XML/Schema, 2014.

[33] GreenStar Network – Building a Zero Carbon Network. http://www.greenstarnetwork.com/, 2015.

[34] LOFAR telescope. http://www.lofar.org/, 2015.

[35] SKA telescope: Square Kilometer Array. https://www.skatelescope.org/sdp, 2015.

[36] SURFsara: High Performance Computing & datainfrastructure for science and industry. https://www.surf.nl/en/about-surf/subsidiaries/surfsara, 2015.

[37] *Algorithms in C third edition Part 5 Graph Algorithms.* Addison-Wesley Professional, Aug. 2001.

[38] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. In *Proceedings of the 37th annual international symposium on Computer architecture(ISCA)*, page 338, New York, New York, USA, 2010.

[39] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab. Hamid, Muhammad Shiraz, Abdullah Yousafzai, and Feng Xia. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications*, 52(0):11 – 25, 2015.

[40] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM.

[41] Ann Arbor, Xi Chen, Ann Arbor, Robert P Dick, Zhuoqing Morley Mao, and Ann Arbor. Performance and Power Modeling in a Multi-Programmed Multi-Core Environment. In *DAC*, pages 1–6, 2010.

[42] Danilo Ardagna, Barbara Panicucci, Marco Trubian, and Li Zhang. Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments. *IEEE Transactions on Services Computing*, 5(1):2–19, January 2012.

[43] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, 2007.

[44] La Barroso and U Hölzle. The case for energy-proportional computing. *IEEE computer*, (December):33–37, 2007.

[45] Aruna Prem Bianzino, Luca Chiaraviglio, and Marco Mellia. Distributed algorithms for green IP networks. *Proceedings - IEEE INFOCOM*, pages 121–126, 2012.

[46] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[47] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In Ian Horrocks and James Hendler, editors, *Proceedings of the first Int'l Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68, Sardinia, Italy, May 2002. Springer Verlag.

[48] T. Brunschwiler, B. Smith, E. Ruetsche, and B. Michel. Toward zero-emission data centers through direct reuse of thermal energy. *IBM J. Res. Dev.*, 53(3):476–488, May 2009.

[49] Ray Carroll, Andrew Mackarel, and Alin Pastrama. Networks and services interconnection between Mantychore and GSN. Technical report, Mantychore Project, 2013.

[50] Rama Chellappa. Gaussian Mixture Models. *Encyclopedia of Biometrics 2009*, (2):659–663, 2009.

[51] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *NSDI'08 Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 337–350, 2008.

[52] Q. Chen, P. Grosso, K. van der Veldt, C. de Laat, R. Hofman, and H. E. Bal. Profiling energy consumption of vms for green cloud computing. In *Proceedings of the International Conference on Cloud and Green Computing (CGC)*, 2011.

[53] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini. An energy saving routing algorithm for a green ospf protocol. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–5, March 2010.

[54] Cisco. Cisco energywise: Power management without borders. Technical report, Cisco, 2012.

[55] Andy Cooke, Alasdair J G Gray, Lisha Ma, Werner Nutt, James Magowan, Manfred Oevers, Paul Taylor, Rob Byrom, Laurence Field, Steve Hicks, Jason Leake, Manish Soni, Antony Wilson, Roney Cordenonsi, Linda Cornwall, Abdeslem Djaoui, and Steve Fisher. R-GMA : An Information Integration System for Grid Monitoring. In *Proceedings of the 11th International Conference on Cooperative Information Systems*, number 1, pages 462—-481, 2003.

[56] O Corcho, P Alper, I Kotsiopoulos, P Missier, S Bechhofer, and C Goble. An overview of S-OGSA: A Reference Semantic Grid Architecture. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):102–115, June 2006.

[57] A. Daouadji and K.-K. Nguyen. Ontology-Based Resource Description and Discovery Framework for Low Carbon Grid Networks. In *2010 First IEEE International Conference on Smart Grid Communications*, pages 477–482, October 2010.

[58] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[59] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. W3C recommendation, W3C, February 2004.

[60] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, D. Klein, and J. Broekstra. The Semantic Web: The roles of XML and RDF. *IEEE Internet Computing*, 15(3):63–74, October 2000.

[61] G. Dhiman and T.S. Rosing. System-Level Power Management Using Online Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):676–689, May 2009.

[62] Gaurav Dhiman. A System for Online Power Prediction in Virtualized Environments Using Gaussian Mixture Models. In *DAC'10*, number 3, pages 807–812, 2010.

[63] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host load prediction in a Google compute cloud with a Bayesian model. *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, November 2012.

[64] G. Dobson, R. Lock, and I. Sommerville. QoSOnt: a QoS Ontology for Service-Centric Systems. In *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 80–87. Ieee, 2005.

[65] Dimitris Economou, Suzanne Rivoire, and Christos Kozyrakis. Full-System Power Analysis and Modeling for Server Environments. In *Workshop on Modeling, Benchmarking, and Simulation (MoBS), held at the International Symposium on Computer Architecture (ISCA)*, 2006.

[66] M. Ellert, M. Grø nager, a. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimäki, O. Smirnova, and a. Wäänänen. Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation Computer Systems*, 23(2):219–240, February 2007.

[67] David Erickson. The Beacon OpenFlow Controller, 2013.

[68] E. Escalona, Shuping Peng, R. Nejabati, D. Simeonidou, J.A. Garcia-Espin, J. Ferrer, S. Figuerola, G. Landi, N. Ciulli, J. Jimenez, B. Belter, Y. Demechenko, C. de Laat, Xiaomin Chen, A. Yukan, S. Soudan, P. Vicat-Blanc, J. Buysse, M. De Leenheer, C. Develder, A. Tzanakaki, P. Robinson, M. Brogle, and T.M. Bohnert. Geysers: A novel architecture for virtualization and co-provisioning of dynamic optical networks and it services. In *Future Network Mobile Summit (FutureNetw), 2011*, pages 1–8, June 2011.

[69] Xiaobo Fan and Wolf-dietrich Weber. Power Provisioning for a Warehouse-sized Computer. In *ISCA '07 Proceedings of the 34th annual international symposium on Computer architecture*, number June, pages 13–23, San Diego, CA, 2007.

[70] Weiwei Fang, Xiangmin Liang, Yantao Sun, and Athanasios V. Vasilakos. Network element scheduling for achieving energy-aware data center networks. *International Journal of Computers, Communications and Control*, 7(2):241–251, 2012.

[71] Mattijs Ghijsen, Jeroen van der Ham, Paola Grosso, Cosmin Dumitru, Hao Zhu, Zhiming Zhao, and Cees de Laat. A semantic-web approach for modeling computing infrastructures. *Computers & Electrical Engineering*, 39(8):2553–2565, November 2013.

[72] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, December 2008.

[73] C. Gunaratne, K. Christensen, and B. Nordman. Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed. *International Journal of Network Management*, 15(5):297–310, September 2005.

[74] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 63–74, New York, NY, USA, 2009. ACM.

[75] A. Hanemann, J. W. Boote, E. Boyd, J. Dur, L. Kudarimoti, Roman Lapacz, D. M. Swany, S. Trocha, and J. Zurawski. Perfsonar: A service oriented architecture for multi-domain network monitoring. In *In Proceedings of the Third International Conference on Service Oriented Computing (ICSOC 2005). ACM Sigsoft and Sigweb*, 2005.

[76] B Hawkins and J Sabin. KairosDB. https://code.google.com/p/kairosdb/.

[77] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters. *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '05*, page 186, 2005.

[78] B. Heller and P. Mahadevan. ElasticTree : Saving Energy in Data Center Networks. In *The 7th USENIX Conference on Network Systems Design and Implementation(NSDI)*, pages 2–17, 2010.

[79] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan and Claypool Publishers, 1st edition, 2009.

[80] Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler. Semantic web architecture: Stack or two towers? In François Fages and Sylvain Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, volume 3703 of *Lecture Notes in Computer Science*, pages 37–41. Springer Berlin Heidelberg, 2005.

[81] Bela Hullar, Laki Sandor, Steger Jozsef, Csabai Istvan, and Vattay Gabor. SONoMA: A Service Oriented Network Measurement Architecture. In *Testbeds and Research Infrastructure. Development of Networks and Communities.*, number May, pages 27–42, 2012.

[82] Hao Jin, Tosmate Cheocherngngarn, Dmita Levy, Alex Smith, Deng Pan, Jason Liu, and Niki Pissinou. Joint host-network optimization for energy-efficient data center networking. *Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013*, pages 623–634, 2013.

[83] Aman Kansal, Feng Zhao, and Arka A Bhattacharya. Virtual Machine Power Metering and Provisioning. In *SoCC'10*, 2010.

[84] Carl Kesselman, Jeffrey M Nick, and Steven Tuecke. The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.

[85] Ricardo Koller. WattApp : An Application Aware Power Meter for Shared Data Centers. In *Proceedings of the 7th international conference on Autonomic computing*, pages 31–40, 2010.

[86] Fanxin Kong and X U E Liu. A Survey on Green-Energy-Aware Power Management for Datacenters. 47(2), 2014.

[87] Ralph Koning, Paola Grosso, and Cees de Laat. Using ontologies for resource description in the CineGrid Exchange. *Future Generation Computer Systems*, 27(7):960–965, July 2011.

[88] Vaid Kushagra. Datacenter Power Efficiency: Separating Fact From Fiction. *In- vited Talk at USENIX HotPower 2010 Report*, 2010.

[89] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Power capping: A prelude to power shifting. *Cluster Computing*, 11(2):183–195, June 2008.

[90] Ricardo Lent. A model for network server performance and power consumption . *Sustainable Computing: Informatics and Systems*, 3(2):80 – 93, 2013.

[91] Dan Li, Yunfei Shang, and Congjie Chen. Software defined green data center network with exclusive routing. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1743–1751, April 2014.

[92] Yanfei Li, Ying Wang, Bo Yin, and Lu Guan. An Online Power Metering Model for Cloud Environment. *2012 IEEE 11th International Symposium on Network Computing and Applications*, pages 175–180, August 2012.

[93] Chia-Hung Lien and Bai Ying-Wen. Web Server Power Estimation, Modeling And management. In *14th IEEE International Conference on Network (ICON '06).*, volume 00, pages 1–6, 2006.

[94] Gongqi Lin, Sieteng Soh, Kwan-Wu Chin, and Mihai Lazarescu. Energy aware two disjoint paths routing. *Journal of Network and Computer Applications*, 43(0):27 – 41, 2014.

[95] Walter Lioen. Jezelf Groen Rekenen met Supercomputers . 2014.

[96] Ruoyan Liu, Huaxi Gu, Xiaoshan Yu, and Xiumei Nian. Distributed flow scheduling in energy-aware data center networks. *IEEE Communications Letters*, 17(4):801–804, 2013.

[97] P. Mahadevan, S. Banerjee, P. Sharma, A. Shah, and P. Ranganathan. On energy efficiency for enterprise and data center networks. *Communications Magazine, IEEE*, 49(8):94–100, August 2011.

[98] P. Mahadevan, S. Banerjee, P. Sharma, a. Shah, and P. Ranganathan. On energy efficiency for enterprise and data center networks. *Communications Magazine, IEEE*, 49(August):94–100, 2011.

[99] P. Mahadevan, P. Sharma, and S. Banerjee. A Power Benchmarking Framework for Network Devices. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, pages 795–808, 2009.

[100] Matthew L Massie, Brent N Chun, and David E Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840, July 2004.

[101] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[102] G. Meijer. Cooling energy-hungry data centers. *Science*, 328(April):318–319, 2010.

[103] Christoph Möbius, Waltenegus Dargie, Senior Member, and Alexander Schill. Power Consumption Estimation Models for Processors , Virtual Machines , and Servers. *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, pages 1–14, 2013.

[104] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, November 1996.

[105] Marcelo R Nascimento, Christian E Rothenberg, Marcos R Salvador, Carlos NA Corrêa, Sidney C de Lucena, and Maurício F Magalhães. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In *Proceedings of the 6th International Conference on Future Internet Technologies*, pages 34–37. ACM, 2011.

[106] Marcelo Ribeiro Nascimento, Christian Esteve Rothenberg, Marcos Rogério Salvador, and Maurício Ferreira Magalhães. Quagflow: partnering quagga with openflow. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 441–442. ACM, 2010.

[107] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 323–336, 2008.

[108] Sung-Kwun Oh and Witold Pedrycz. The design of self-organizing Polynomial Neural Networks. *Information Sciences*, 141(3-4):237–258, April 2002.

[109] Asif Osman, Ashiq Anjum, Naheed Batool, and Richard Mcclatchey. A Fault Tolerant , Dynamic and Low Latency BDII Architecture for Grids. 3(4):1–18, 2010.

[110] Leonardo Piga, ReinaldoA. Bergamaschi, and Sandro Rigo. Empirical and analytical approaches for web server power modeling. *Cluster Computing*, pages 1–15, 2014.

[111] E. Prudhommeaux and A. Seaborne. SPARQL Query Language for RDF. *W3C Recommendation*, 4:1–106, 2008.

[112] Karthick Rajamani, Heather Hanson, Juan Rubio, Soraya Ghiasi, and Freeman Rawson. Application-Aware Power Management. *2006 IEEE International Symposium on Workload Characterization*, pages 39–48, October 2006.

[113] Charles Reiss, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. Heterogeneity and Dynamicity of Clouds at Scale : Google Trace Analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, pages 7:1–7:13, 2012.

[114] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A Comparison of High-Level Full-System Power Models. In *Workshop on Power Aware Computing and Systems (HotPower), held at the Symposium on Operating Systems Design and Implementation (OSDI)*, 2008.

[115] Mirza Pahlevi Said and Isao Kojima. S-MDS: Semantic Monitoring and Discovery System for the Grid. *Journal of Grid Computing*, 7(2):205–224, November 2008.

[116] Jennifer M Schopf, Ioan Raicu, Laura Pearlman, Neill Miller, Carl Kesselman, and Mike D Arcy. Monitoring and Discovery in a Web Services Framework : Functionality and Performance of Globus Toolkit MDS4. Technical report, 2005.

[117] S. Selvi, R.a. Balachandar, K. Vijayakumar, N. Mohanram, M. Vandana, and Rajagopalan Raman. Semantic Discovery of Grid Services Using Functionality Based Matchmaking Algorithm. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 170–173. Ieee, December 2006.

[118] Y. Shang, D. Li, and M. Xu. A Comparison Study of Energy Proportionality of Data Center Network Architectures. In *32nd International Conference on Distributed Computing Systems Workshops*, pages 1–7, June 2012.

[119] Shuaiwen Song, Chunyi Su, Barry Rountree, and Kirk W. Cameron. A simplified and accurate model of power-performance efficiency on emergent GPU architectures. *Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013*, pages 673–686, 2013.

[120] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy Aware Consolidation for Cloud Computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, 2008.

[121] Robert Stanforth, Evgueni Kolossov, and Boris Mirkin. Hybrid k-Means: Combining Regression-Wise and Centroid-Based Criteria for QSAR. In *Selected Contributions in Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 225–233. Springer Berlin Heidelberg, 2007.

[122] Joszef Steger, Sandor Laki, and Peter Matray. A Monitoring Framework for Federated Virtualized Infrastructures. *Measurement-based experimental research: methodology, experiments and tools*, 7586, 2012.

[123] Nguyen Huu Thanh and Pham Ngoc Nam. Enabling Experiments for Energy-Efficient Data Center Networks on OpenFlow-based Platform. (2):239–244, 2012.

[124] Nguyen Huu Thanh, Pham Ngoc Nam, Truong Thu Huong, Tran Ngoc Thuan, Nguyen Minh Duong, Giang Nguyen Van, Nguyen Tai Hung, Ngo Quynh Thu, Hock David, and Schwartz Christian. Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks. *Simulation Modelling Practice and Theory*, 39(0):20 – 40, 2013. S.I.Energy efficiency in grids and clouds.

[125] Jeroen J. van der Ham, Freek Dijkstra, Franco Travostino, Hubertus M.A. Andree, and Cees T.A.M. de Laat. Using {RDF} to describe networks. *Future Generation Computer Systems*, 22(8):862 – 867, 2006.

[126] Karel van der Veldt, Inder Monga, Jon Dugan, Cees de Laat, and Paola Grosso. Carbon-aware path provisioning for NRENs. In *2014 International Green Computing Conference (IGCC)*, Dallas TX USA, 3-5, Nov. 2014.

[127] Ingolf Wabmann, Daniel Versick, and Djamshid Tavangarian. Energy consumption estimation of virtual machines. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, page 1151, 2013.

[128] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. CARPO: Correlation-aware power optimization in data center networks. In *Proceedings IEEE INFOCOM*, pages 1125–1133, March 2012.

[129] Zhengkai Wu, Junyao Zhang, C. Giles, and Jun Wang. Classified power capping with distribution trees in cloud computing. In *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*, pages 319–328, July 2011.

[130] Wei Xing, Oscar Corcho, Carole Goble, and Marios D. Dikaiakos. An ActOn-based semantic information service for Grids. *Future Generation Computer Systems*, 26(3):324–336, March 2010.

[131] Mingwei Xu, Yunfei Shang, Dan Li, and Xin Wang. Greening data center networks with throughput-guaranteed power-aware routing. *Computer Networks*, 57(15):2880–2899, October 2013.

[132] A. Zafeiropoulos. Baseline studies of an NREN and the whole GÉANT network. In *, TERENA Green-Workshop 2012*, Utrecht, Netherlands, 2012.

[133] Reza Zamani and Ahmad Afsahi. A Study of Hardware Performance Monitoring Counter Selection in Power Modeling of Computing Systems. In *The 2nd International Workshop on Power Measurement and Profiling (PMP 2012)*, 2012.

[134] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L. Hellerstein. Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments. In *Proceedings of the 9th international conference on Autonomic computing*, pages 145–154, 2012.

[135] Z. Zhao, P. Grosso, J. van der Ham, R. Koning, and C. de Laat. An agent based network resource planner for workflow applications. *International Journal of Multiagent and Grid Systems*, 7/6:187–202, 2011.

[136] Kuangyu Zheng, Xiaodong Wang, Li Li, and Xiaorui Wang. Joint Power Optimization of Data Center Network and Servers with Correlation Analysis. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014.

[137] Hao Zhu, K. Van Der Veldt, P. Grosso, Xiangke Liao, and C. De Laat. Ekb: Semantic information system for energy-aware monitoring in distributed infrastructures. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 60–67, 2013.

[138] Hao Zhu, Karel van der Veldt, Paola Grosso, and Cees de Laat. EDL: an energy-aware semantic model for large-scale infrastructures. Technical report uva-sne, no. 2014-02, University of Amsterdam, February 2014.

[139] Hao Zhu, Karel van der Veldt, Zhiming Zhao, Paola Grosso, Dimitar Pavlov, Joris Soeurt, Xiangke Liao, and Cees de Laat. A semantic enhanced power budget calculator for distributed computing using ieee 802.3az. *Cluster Computing*, 18(1):61–77, 2015.

Publications

Journals

[1] **H. Zhu**, K. van der Veldt, Z. Zhao, P. Grosso, D. Pavlov, J. Soeurt, X. Liao, and C. de Laat (2015). A semantic enhanced power budget calculator for distributed computing using ieee 802.3az, Cluster Computing, 18(1), 61-77, 2015.

[2] **H. Zhu**, X. Liao, C. de Laat and P. Grosso (2015). Joint flow routing-scheduling for energy efficient software defined data center networks. Journal of Network and Computer Applications. (under revision)

[3] **H. Zhu**, X. Liao, C. de Laat and P. Grosso (2015). Evaluation of non-linear approaches for power estimation in a computing cluster. Sustainable Computing: Informatics and Systems. (invited paper, under revision)

[4] M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, **H. Zhu**, Z. Zhao and C. de Laat (2013). A semantic-web approach for modeling computing infrastructures. Computers and Electrical Engineering, 39 (8), 2553-2565.

Conferences

[1] **H. Zhu**, J. Aznar Baranda, C. de Laat and P. Grosso (2015). Green routing for software defined data center networks based on OpenNaaS. 4th International Conference on Green IT Solutions (ICGreen), Milan, Italy.

[2] **H. Zhu**, P. Grosso, X. Liao and C. de Laat (2014). Evaluation of approaches for power estimation in a computing cluster. In 2014 International Green Computing Conference (IGCC), pages 1-10, Dallas, TX.

[3] **H. Zhu**, K. van der Veldt, P. Grosso, X. Liao and C. de Laat (2013). EKB: a semantic information system for energy-aware monitoring in distributed infrastructures. In Proceedings: 2013 IEEE Third International Conference on Cloud and Green Computing: CGC 2013, pages 60-67, Karlsruhe, Germany.

[4] **H. Zhu**, K. van der Veldt, P. Grosso, Z. Zhao, X. Liao and C. de Laat (2012). Energy-aware semantic modeling in large scale infrastructures. In Work in Progress Sessions (WiP): 2012 IEEE International Conference on Green Computing and Communications (GreenCom 2012): pages 11-14, Besancon, France.

Technical Reports

[1] **H. Zhu**, K. van der Veldt, P. Grosso and C. de Laat (2014). EDL: an energy-aware semantic model for large-scale infrastructures. (Technical report UVA-SNE, no. 2014-02). Amsterdam: Universiteit van Amsterdam, System and Network Engineering.

[2] M. Ghijsen, J. van der Ham, P. Grosso, C. Dumitru, **H. Zhu**, Z. Zhao and C. de Laat (2013). A semantic-web approach for modeling computing infrastructures. (SNE technical report, no. 2013-01). Amsterdam: Universiteit van Amsterdam, System and Network Engineering.

Acknowledgements

I would like to acknowledge those people who helped me practically or mentally pursue my PhD at University of Amsterdam.
....

Hao Zhu
Amsterdam, July 2015