Enabling Operational Data Analytics for Datacenters through Ontologies, Monitoring, and Simulation-based Prediction

Shekhar Suman [+]
s.suman@student.vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Xiaoyu Chu [+]
x.chu@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Dante Niewenhuis
d.niewenhuis@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Sacheendra Talluri
s.talluri@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Tiziano De Matteis
t.de.matteis@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

Alexandru Iosup
a.iosup@vu.nl
Vrije Universiteit Amsterdam
The Netherlands

ABSTRACT

Datacenters are key components in the ICT infrastructure supporting our digital society. Datacenter operations are hampered by operational complexity and dynamics, risking to reduce or even offset the performance, energy efficiency, and other datacenter benefits. A promising emerging technology, Operational Data Analytics (ODA), promises to collect and use monitoring data to improve datacenter operations. However, it is challenging to organize, share, and leverage the massive and heterogeneous data resulting from monitoring datacenters. Addressing this combined challenge, starting from the idea that graphs could provide a good abstraction, in this work we present our early work on designing and implementing a graph-based approach for datacenter ODA. We focus on two main components of datacenter ODA. First, we design, implement, and validate a *graph-based ontology for datacenters* that captures both high-level meta-data information and low-level metrics of operational data collected from real-world datacenters, and maps them to a graph structure for better organization and further use. Second, we design and implement *ODAbler, a software framework for datacenter ODA*, which combines ODA data with an online simulator to make predictions about current operational decisions and other what-if scenarios. We take the first steps to illustrate the practical use of ODAbler, and explore its potential to support datacenter ODA through graph-based analysis. Our work helps construct the case that graph-based ontologies have great value for datacenter ODA and, further, to improving datacenter operations.

CCS CONCEPTS

• **Computer systems organization → Maintainability and maintenance**.

KEYWORDS

graph-based ontology, ODAbler, OpenDC, operational data analytics, monitoring, mapping, analysis, simulation, datacenter

[+]These two authors contributed equally to this work.

ICPE '24 Companion, May 7–11, 2024, London, United Kingdom
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0445-1/24/05
https://doi.org/10.1145/3629527.3652897

ACM Reference Format:
Shekhar Suman [+], Xiaoyu Chu [+], Dante Niewenhuis, Sacheendra Talluri, Tiziano De Matteis, and Alexandru Iosup. 2024. Enabling Operational Data Analytics for Datacenters through Ontologies, Monitoring, and Simulation-based Prediction. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629527.3652897

1 INTRODUCTION

Our economy, academia, and more broadly our society rely on ICT infrastructures; for example, in the Netherlands, nearly two-thirds of the $ 1 trillion GDP, over 3 million jobs, and a large fraction of economic growth depend directly on such infrastructure [6]. Datacenters are one of the most important components of the modern ICT infrastructure. The complexity of modern datacenters data introduces significant operational challenges, including challenges related to performance, availability, and efficient use of energy. To address such challenges, we need new ways to collect, share, and understand operational data across different operational layers of datacenters, simplifying, hardware, software, and applications. In this work, we consider how to enable *Operational Data Analytics (ODA)*, a family of concepts and techniques that leverage monitoring data to extract high-level, actionable knowledge that can be used to drive operational decisions [12]. We focus in this work on how ontologies, monitoring, and simulation-based prediction can enable ODA for datacenters.

Figure 1 shows a common ODA process, derived from Netti et al. [12]. The process includes five components: (*1*) The physical *Datacenter*, which contains any kind of hardware, energy-transferring devices, and cooling infrastructure; (*2*) *Data Collection*, which includes different sources of operational data, such as live monitoring sensors, tracing frameworks, and logging, (*3*) *Data Storage*, where, in this work, we propose to augment the traditional time-series database with an ontology-based approach, (*4*) *Data Analytics*, where different techniques, such as workload characterization and modeling, are used and possibly combined, to analyze and optimize datacenter operations, and, last, (*5*) *Reassessment and Redeployment*, where past analytics results and current status are used to drive decisions and optimize the ODA process automatically.

We augment in this work the typical stage 3 in ODA processes with an ontology-based approach, and stage 4 with an approach that leverages the ontology and combines analytical capabilities

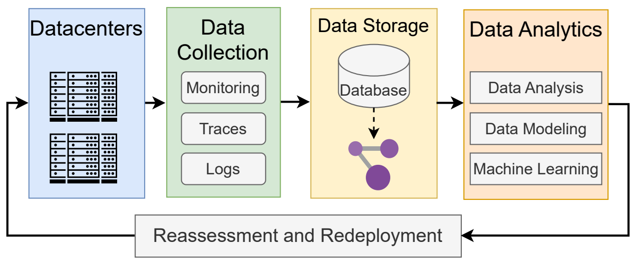

Figure 1: ODA process.

related to it into a larger ODA framework, ODAbler. Our motivation to develop a datacenter ontology are based on the promise of graph data for any field [14]: (1) Ontology can structure and formalize complex hierarchies and relationships in graph formats and thus can be a good way to organize operational data collected from datacenters, (2) Many powerful graph-based algorithm applications can be used for ODA data, leading to insights that are difficult to obtain from non-graph data and in particular from mere time-series, and (3) From a computer systems perspective, we also aim to give insights into building graph-based datasets and data management approaches for datacenter operational data, encouraging future research in this area.

Several studies have already developed and built ontologies specifically for ICT infrastructures and datacenters. [1–3]. The CloudLightning Ontology [1] is designed to address the heterogeneous resources management interoperability issues. The HPC ontology [9] is used for managing training datasets of AI models for addressing various challenges in HPC. The ICT Infrastructures ontology network [2] is a high-level datacenter ontology that includes software, database, hardware, server, and network. However, these ontologies are neither built on the monitoring data, nor do they cover all levels and attributes of datacenters, and they are not designed for operational data analytics, which requires special functions. Thus, a data-centered ontology for operational data analytics is still lacking.

Addressing a key gap around the use of ontology-based approaches for datacenter ODA, in this work we take first steps toward a universal framework for datacenter ODA, with a twofold contribution:

(1) We design and implement an ontology to structure the data collected in datacenters (in Section 3). It is the first data-centered ontology for datacenters, which could enable complex graph analysis and applications in the future. We validate the implementation through a prototype ontology that meets the requirements for a real-world HPC cluster dataset.
(2) We design and implement the *ODAbler* framework for datacenter ODA framework (in Section 4). Based on the ontology designed for this work, ODAbler enables the ingestion and export of operational metrics typical of datacenters. It also supports complex analysis of datacenter scenarios, around a state-of-the-art simulator (here, OpenDC [11]). Last, it offers the technology framework necessary to explore, in the future, the use of graph-based analysis to understand how datacenters operate and to conduct what-if analysis for datacenters in a new way.

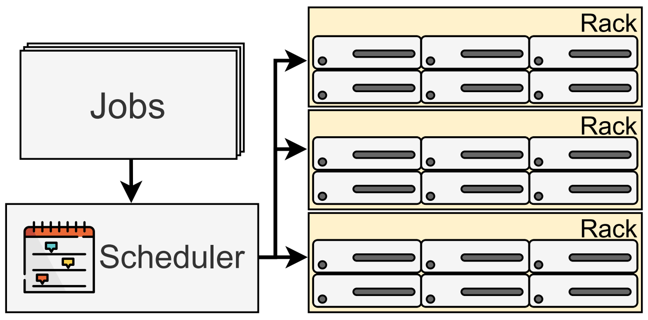

Figure 2: System model.

2 BACKGROUND

In this section, we describe the system model from which we collected data, and provide basic information about ontology.

2.1 System model

Figure 2 depicts the system model in the HPC cluster we used to collect data. A datacenter system is composed of many racks, each of which accommodates multiple server nodes. The nodes are connected through a network interconnect. Different kinds of jobs are submitted to a scheduler which then schedules them onto the nodes of the datacenter. A job can use a single node or multiple nodes.

We build the ontology based on the data collected from the *SURF Lisa cluster* by two tools. First, *SLURM* scheduler logs, from which we collect 10 months of job data, from the end of December 2021 to November 2022, with 1,596,963 records and 16 metrics. The dataset includes job-related metrics such as the number of machines allocated to jobs, the nodes that were allocated, and the completion status. Second, *Prometheus* monitor, from which we collect 5 months of node data, from June 2022 to November 2022, with 127,827,719 records and 82 metrics. This dataset includes node-related metrics such as node capacities, CPU load, memory, network, and power and temperature metrics. All these metrics are supposed to be covered in the designed ontology.

2.2 Ontology and OWL basics

Ontology is the field of science that helps us investigate what types of *entities* (or *classes*, sometimes also called *concepts*) exist in a domain of discourse, how they are grouped into categories, and how they are related to one another on the most fundamental level. An ontology along with a set of individual *instances* of classes constitutes a *knowledge base* [13]. The common reasons for developing an ontology are below [13]: (1) Sharing a common understanding of the structure of information among the utilizing resources; (2) Enabling reusability of domain knowledge; (3) Making explicit domain assumptions; (4) Analysing domain knowledge.

The *Web Ontology Language (OWL)* is a formal language for expressing ontologies and is based on the *description logic (DL)*. The underlying format that is fundamental to storing a wide range of information in OWL-based ontologies is the *Resource Description*

Framework (RDF). The information stored in an RDF consists of a triple: *subject, property, and object.* For instance, *("DataCenter-XYZ", hasAmbientTemperature, "21 degrees")* states that a *DataCenter-XYZ* has an ambient temperature of *21 degrees* (which could further be explicitly related to the time of record capture using another property, or annotated with the timestamp information).

In this project, OWL has been used for modeling the ontology (where OWL support is made available in the Python platform by *Owlready2* [8]), and a few experiments (using the *SPARQL* language) have been conducted to validate that the designed ontology meets the expectations by satisfying the requirements outlined in the next section. Some of the key definitions that set up the basis of the ontology modeling using OWL are outlined below:

- *Classes*: Entities in an object-oriented world. In an ontology, an individual can belong to several classes. The relation between the two classes can be *disjoint* (two disjoint classes cannot have individuals in common), *pairwise disjoint* (any pair made up of two classes from this list are disjoint), and *partitions* (to declare "either-or" of a class).
- *Data properties*: properties whose values are data (number, text, date, boolean, etc.). The data properties have a *domain* (the class for which the property is defined), and *range* (the associated datatype, which can be an integer or a real number, boolean, character string, date, and so on).
- *Object properties*: properties whose values are entities (i.e. ontology individuals). The range of object properties is the class of associated objects.

Generally, developing an ontology includes the following practical steps [13]:

1. Determine the domain and scope of the ontology.
2. Consider reusing existing ontologies.
3. Enumerate important terms in the ontology.
4. Define the classes and the class hierarchy.
5. Define the data properties of classes.
6. Define the object properties of classes.
7. Create instances.

3 ONTOLOGY DESIGN

In this section, we first analyze the requirements for the proposed datacenter ontology. Following the requirements, we design and implement the ontology according to the metrics collected from an HPC datacenter, both high-level and detailed. We also explore how they could be useful for Operational Data Analytics (ODA). Finally, we conduct validations to ensure that the ontology aligns with the proposed requirements.

Following the ontological process introduced in Section 2.2, we determine the domain and scope of the ontology through requirements analysis in Section 3.1 as step 1. We reuse the existing ontology in Section 3.2 as step 2. We do high-level design including identifying important terms in Section 3.3 as step 3. We cover steps 4-6 in in Section 3.4, and create instances for validation in Section 3.5.

3.1 Requirements analysis

In order to support operational data analytics, the datacenter ontology has to cover a large scope, including infrastructure, hardware, software, and applications. Besides, we give four functional requirements and two non-functional requirements for the datacenter ontology.

Table 1: The overall statistics of the designed ontology, based on metrics from the SURF Lisa dataset.

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

3.1.1 *Functional requirements.*

FR1. Time-series modeling. The ontology should support the modeling of attributes extracted from time-series metrics collected from a large-scale computing infrastructure, e.g., an HPC cluster.

FR2. Resource description. The ontology should describe the datacenter cluster resources in a structured way, including details about nodes, processors, etc.

FR3. Performance metrics. The ontology should capture and analyze performance metrics such as resource utilization and energy consumption.

FR4. Consistency and accuracy. The ontology should be consistent and accurate in its representation by reflecting the state of the datacenter and its resources.

3.1.2 *Non-functional requirements.*

NFR1. Interoperability. The ontology should be designed with interoperability in mind, which should facilitate integration or reuse with/by other ontologies.

NFR2. Usability. The ontology should be user-friendly, having sufficient comments or labels for accessibility by both experts and non-experts.

3.2 Reusing existing ontologies

The best resource that is closer to our ontology requirements is a work of literature studying the ontological representation of time-series observations on the Semantic Sensor Web [4]. It suggests the usage of three important modeling classes, out of which we find that the most relevant class *"Observation"* can be reused, and thus discussed in detail below. The two other classes *"Observation-Collection"* and *"TimeSeriesObservation"* do not seem convincingly reusable, mainly because of the nature of the requirements to model a sample time-series data. If we were to model multiple time-series data in the ontology, then we could have inherited the same structure. But, in this project, we limit ourselves to demonstrating a single set of records in the ontology model, other than the established relations amongst the classes. Some of the relationships for observations that have been used here are listed below:

- *featureOfInterest*: Representation of the object being observed.
- *observedProperty*: The phenomenon for which the observation result provides an estimate of its value.

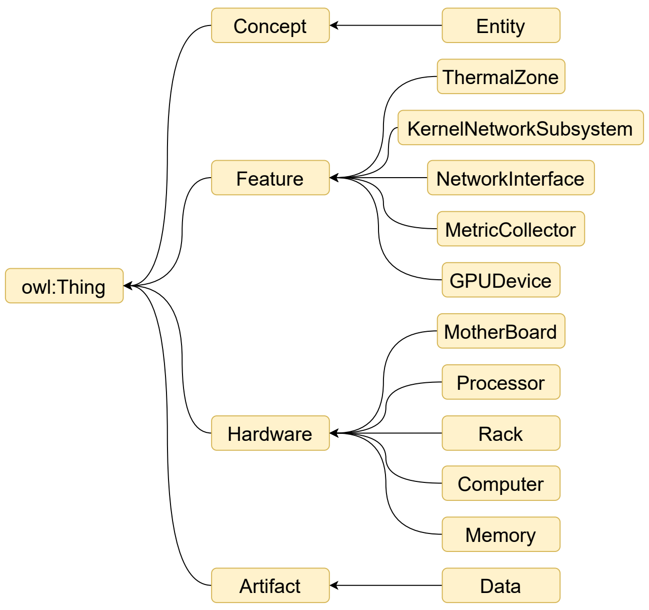

Figure 3: High-level classes of the datacenter ontology. The arrow denote the "is-a" relation between classes.

- *samplingTime* or *generatedAtTime*: The time when the phenomenon was measured.
- *result* or *value*: An estimate of the value of a property generated using a known procedure.
- *memberOf*: A relation to a set of observations of observation collection.

3.3 High-level design and classes

The ontology has been developed to structure and formalize complex hierarchies and relationships of operational data in datacenters, it provides a foundation to enable further graph-based applications. We first give an overview of the statistics of the designed ontology, then we describe the high-level classes, and the key subclasses step by step.

The ontology has been developed in the OWL language by using *Protégé*[1] as the ontology editor, knowledge management, and visualization system. We follow a common naming convention when defining the ontology terms: Singular nouns in CamelCase are used to present a Class, while Property names start with lowercase letters. The statistic of the implemented prototype is shown in Table 1. The ontology has 82 classes, 17 object properties, and 63 data properties in total.

3.3.1 High-level classes. Figure 3 depicts the high-level classes in the designed ontology. It consists of: a *Concept* class, which describes the *Entity* in datacenter, such as *Software* entity, which includes *JobScheduler, MetricsExporter, ResourceManager, MonitoringSystem* and so on; a *Feature* class describing different metric collector in the datacenter; a *Hardware* class, which defining hardware configurations such as *Processor, Memory*; a *Artifact* class, which captures various source of *Data* from *Job* or *MonitoringMetrics*.

[1]Protege - https://protege.stanford.edu/

Table 2: Object properties of the datacenter ontology.

Property	Domain	Range
atLocation	Thing	Thing
exportMonitoringMetrics	MetricsExporter	MonitoringMetrics
featureOfInterest	Thing	Feature
hasExitCode	Thing	ExitCode
hasJobScheduler	Computer	JobScheduler
hasMember	Thing	Thing
hasMetricsExporter	Entity	MetricsExporter
hasMonitoringSystem	Computer	MonitoringSystem
hasResourceManager	Computer	ResourceManager
isScheduledOn	Data	Thing
isScheduledOnServer	Thing	Server
manageJob	Software	Job
managesJob	JobScheduler	Job
measuresValueOfThing	MonitoringMetrics	Thing
hasMember	Thing	Thing
observedProperty	Thing	Property

3.3.2 Key subclasses. *MonitoringMetrics* is an important subclass of class *Data*, where we map the metrics collected in the dataset to the ontology. Here we capture different kinds of metrics of datacenter including energy-related data such as *Temperature, EnergyUsage*, and resource-related data such as *CPULoadAverage, NumberOfIOs*. *MetricCollector* is a subclass of class *Feature*, which shows different data collectors such as *CPULoadCollector, MemoryStatisticsCollector*. The subclasses of class *Hardware* shows the common components of a datacenter, including *Server, Rack, Processor* etc.

3.4 Detailed design and propertites

There are two kinds of properties: *object properties* are whose values are entities (i.e., ontology individuals), *data properties* are whose values are data (numbers, texts, dates, Booleans, etc.). In this subsection, we will introduce the details of object and data properties in our ontology.

3.4.1 Object Properties. Object properties indicate the relationships between two classes. Table 2 shows the information of object properties in the designed ontology. The class *Thing* has eight relationships with other classes, including: *atLocation*, presents *Thing* is at some location; *featureOfInterest*, which describes the feature being observed; *hasExitCode*; *hasMember*, indicates the membership relation between two entities, and *memberOf* is an inverse relation of it; *isScheduledOnServer*, indicates job is scheduled on some server (node); *observedProperty*, which provides an estimation of observed value. The class *Computer* has three relations: *hasJobScheduler, hasMonitoringSystem, hasResourceManager*, which reveals the relations between hardware and software. The object properties show the complexity of hierarchies in different layers inside a datacenter, so it is the key to linking different components and metrics.

3.4.2 Data Properties. Figure 4 shows partial data properties in this ontology. Since the goal of ontology is to better organize the operational data generated in datacenter, it should cover time-series metrics as required. The data properties include all the metrics we can collect in the SURF Lisa dataset, and the range of these properties is either float or string.

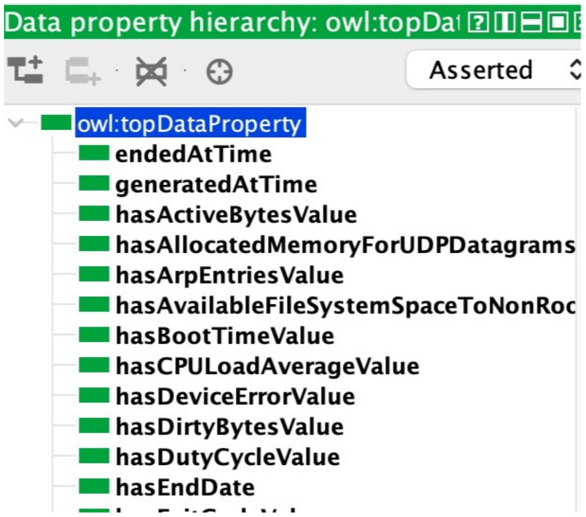
Data property hierarchy: owl:topDat
Asserted
owl:topDataProperty
- endedAtTime
- generatedAtTime
- hasActiveBytesValue
- hasAllocatedMemoryForUDPDatagrams
- hasArpEntriesValue
- hasAvailableFileSystemSpaceToNonRoc
- hasBootTimeValue
- hasCPULoadAverageValue
- hasDeviceErrorValue
- hasDirtyBytesValue
- hasDutyCycleValue
- hasEndDate

Figure 4: Partial data properties of the datacenter ontology.

3.5 Validation

This section is to validate that the modeled ontologies meet the requirements of an ontology in terms of structural modeling. It should be noted that the overall ontology structure could be validated to be syntactically correct using the graph dump property (*ontology.graph.dump* in Python's *Owlready2* module[2]), to ensure that there are no errors encountered while representing data with the ontology (e.g., datatype mismatch, incorrect attribute-name, data missing scenario). If the modeled ontology is structurally incorrect while adding data, then the graph dump statement should give an error.

3.5.1 Validation of key properties. Validation of some of the key properties that are common in both ontologies (e.g. ambient temperature, host power usage, etc.). We perform a SPARQL query to validate that the results are matched in both ontologies, as shown in the listing below:

```
result_surf = list(default_world.sparql("""
    PREFIX  <https://example.org/hpcontology_surf.owl#>
    SELECT DISTINCT ?x where{
    ?x rdfs:subClassOf* Property .
    }
    """))
```

3.5.2 Validation of graph structure. Verification that the modeled ontology also reflects the graphical layout, which could be inferred to create general graph structures on the fly for analytical purposes (e.g., for performing ODA-related analysis). The goal of this experiment is to verify that the ontology modeled for both HPC clusters can be visualized in the form of graphical layouts with nodes and edges, which could be easily converted to a graphical structure in a graph database using available tools. The resulting graphical layout for SURF's LISA cluster is visualized using *WebVOWL* ontology visualization and shown in Figure 5, from which we can see properties such as *NodeTemperature, PowerUsage, FileSystemSize* are presented in a graph format, with the relations to other metrics in the ontology.

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

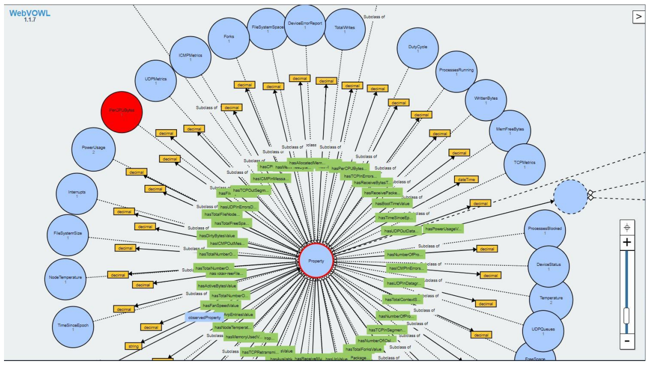

Figure 5: SURF's OWL ontology visualization (using WebVOWL) showing the graph layout according to the VOWL specification, and listing graphical nodes and edges information on the right side information box.

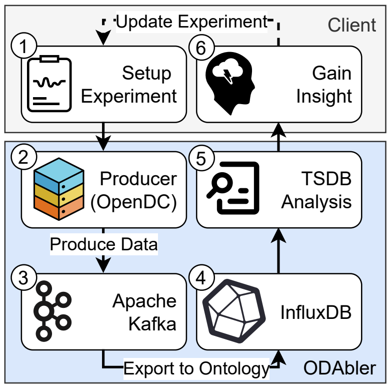

Figure 6: High-level architecture of ODAbler framework.

4 ODABLER: DESIGN OF ONTOLOGY-BASED SIMULATION IN OPENDC

We propose and implement ODAbler as a prototype to show how the ontology can benefit operational data analytic.

4.1 Architecture of ODAbler framework

The key design element of ODAbler involves several key elements as represented in Fig. 6. Some of the key elements include OpenDC itself, and *Apache Kafka* as the middleware which is responsible for message publishing to a time-series database *InfluxDB*, using the *Telegraf* agent. Once the time-series data is available at *InfluxDB*, the ODAbler client application performs out-of-band analysis (as part of the current scope of its implementation)

4.2 Implementation of the ODAbler framework

4.2.1 Ontology-driven export of OpenDC metrics. As shown in Table 3, the corresponding attributes or properties are selected to be exported to the time-series database *InfluXDB* for further analysis by the ODAbler analysis client. This screenshot is taken from the OpenDC's *MonitoringMetrics* class, which is a data class whose sole purpose is to hold data of various metrics as reflected by the name of the member properties.

Table 3: Metrics exported to time-series database reusing the concepts from the designed ontology in the section 3.

Name	Type
cpuIdleTime	Long
cpuActiveTime	Long
cpuLostTime	Long
energyUsage	Double
upTime	Long
serverId	Int
timestamp	Long
cpuUtilization	Double
powerUsage	Double
guestsRunning	Int
policyId	Int

4.2.2 Technical implementation. The technical implementation of the ODAbler framework involves the following steps: (1) Enable fault injection in OpenDC; (2) Launch InfluxDB and Kafka; (3) Start Telegraf service; (4) Launch OpenDC server, then launch ODAbler client analyzer application. (5) Export operational data from OpenDC to Apache Kafka. (6) Kafka exports the ontology-driven relevant power usage and energy usage metrics (besides others) to InfluxDB via the Telegraf agent. (7) ODAbler performs out-of-band analysis on InfluxDB data, once the data is fully available.

4.3 Exploration of graph applications

We explore the potential applications of ontology-based graph applications for datacenter operational data analytics. (We have not yet built these capabilities in ODAbler, but plan to do so.)

4.3.1 Graph queries. Application performance and behavior are linked across the hardware-software stack. However, the metrics are isolated in different parts of the stack. Ontology-enabled monitoring and analysis tools help link metrics across the stack. Listing 1 provides an example query to access all machines running an application using the "production" database and the p99 latency of the connection between the application and the database.

```
SELECT ?appname , ?machine , p99(?latency)
WHERE
  {
    ?x    appname       ?appname ;
          isScheduledOn ?machine ;
          linkedTo      ?link .
    ?link linkedTo      ?db .
          hasMetric     ?y .
    ?y    metricname    "latency" .
          value         ?latency ;
    ?db  appname        "production" .
  }
```

Listing 1: Example query to retrieve p99 latency of all apps connected to the "production" database.

4.3.2 Graph analysis. The typical data center data analysis workflow now involves manually collating and analyzing metrics data. New databases [7] have demonstrated the benefit of graph-aware query engines for linked data. However, the link information is unique to each data center, hindering the development of ODA-specific databases and limiting us to slow ad-hoc data analysis. A common ontology would allow data center operators to bring powerful tools to bear on operational data analytics [12] and workload modeling [15].

4.3.3 Graph machine learning. Machine learning has proven promising in data center resource management applications [10, 15]. However, automatically enhancing datacenter processes remains a challenge. Data availability and domain shift are two obstacles to pervasive machine learning in the data center. Each datacenter has its own idiosyncratic data collection architecture, and ML systems trained on one datacenter's data do not prove helpful in other datacenters. Data normalized using a common ontology helps tackle these obstacles.

5 RELATED WORK

An ontology encompasses a representation, formal naming, and definition of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains of discourse. The basic idea is to represent the properties of a subject area and their relationships, by defining a set of concepts and categories that represent the subject. One of the main reasons for designing an ontology for HPC is to make training datasets and AI models FAIR (FAIR data principles describe Findability, Accessibility, Interoperability, and Reusability) [9]. Some of the existing HPC ontology design already captures both high-level meta information as well as low-level data content for software, hardware, experiments, workflows, training datasets, AI models, and so on [3].

HPC ontology modelling work has already been done in previous scientific research. There are several research works already done in the field of HPC ontology, for example, by C. Liao et al. [9], and by Castañé et al. [1], amongst others. One of the comprehensive HPC ontology designs that already captures both high-level meta information as well as low-level data content for software, hardware, experiments, workflows, training datasets, AI models, and so on is available as HPC Ontology. There are other works of literature on HPC resources' ontology models like Zhou et al. [19], Zhao et al. [18], Tenschert [16] and others, but those are simplified where the main goal of the authors has been to decompose applications between compute and data processes for HPC environments. There are several other works of literature presenting a unified ontology of cloud computing like Youseff et al. [17] and Imam [5]. Amongst these works, we did not find any literature studying the modelling of ontology derived from the metrics collected from operational HPC clusters. But, these metrics are the source of ODA framework design, and, so, the ODA framework should be driven by the modelled ontology, which should be derived from those captured metrics of large-scale computing infrastructures.

6 CONCLUSION AND FUTURE WORK

This paper presents our ongoing efforts to build a datacenter ontology to enable operational data analytics based on the data from a real-world HPC cluster. We adopt both high-level and detailed designs to cover the designed requirements. The resulting datacenter ontology has modeled properties of essential concepts of this domain, including hardware, software, and collected metrics.

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

Through the validation, the ontology can support typical search queries using SPARQL.

An essential item of future work is to incorporate graph-based analytics into ODAbler, and, further, explore applications of this technology. Although the exact capabilities of a graph-based ODAbler, and more generally of graph-based datacenter ODA, are largely unknown, this line of future work will bring evidence of whether our exploration in Section 4.3 is correct and could result in a new way of understanding datacenters.

Future work will also include extensions of the current ontology, such as more comprehensive relations between different entities. We will also add more individuals to the ontology and conduct graph-based experiments to see if the ontology can help better understand the datacenter operation. The current draft ontology is available online at *https://github.com/am-i-helpful/hpc-ontology-modeller*.

ACKNOWLEDGEMENT

We thank the Dutch National Supercomputing Center SURF for providing the data. We thank the China Scholarship Council (CSC) for supporting Xiaoyu Chu. We thank the support of Netherlands-funded projects NWO OffSense and GFP 6G FNS, and EU-funded projects MCSA-RISE Cloudstars and Horizon Graph-Massivizer.

REFERENCES

[1] Gabriel G. Castañé, Huanhuan Xiong, Dapeng Dong, and John P. Morrison. 2018. An ontology for heterogeneous resources management interoperability and HPC in the cloud. *Future Gener. Comput. Syst.* 88 (2018), 373–384. https://doi.org/10.1016/j.future.2018.05.086

[2] Óscar Corcho, David Chaves-Fraga, Jhon Toledo, Julián Arenas-Guerrero, Carlos Badenes-Olmedo, Mingxue Wang, Hu Peng, Nicholas Burrett, Jose Mora, and Puchao Zhang. 2021. A High-Level Ontology Network for ICT Infrastructures. In *The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12922)*, Andreas Hotho, Eva Blomqvist, Stefan Dietze, Achille Fokoue, Ying Ding, Payam M. Barnaghi, Armin Haller, Mauro Dragoni, and Harith Alani (Eds.). Springer, 446–462. https://doi.org/10.1007/978-3-030-88361-4_26

[3] Yu Deng, Ronnie Sarkar, HariGovind V. Ramasamy, Rafah Hosn, and Ruchi Mahindru. 2013. An Ontology-Based Framework for Model-Driven Analysis of Situations in Data Centers. In *2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*. IEEE Computer Society, 288–295. https://doi.org/10.1109/SCC.2013.98

[4] Cory Andrew Henson, Holger Neuhaus, Amit P Sheth, Krishnaprasad Thirunarayan, and Rajkumar Buyya. 2009. An ontological representation of time series observations on the semantic sensor web. (2009).

[5] Fahim T Imam. 2016. Application of ontologies in cloud computing: The state-of-the-art. *arXiv preprint arXiv:1610.02333* (2016).

[6] Alexandru Iosup, Fernando Kuipers, Ana Lucia Varbanescu, Paola Grosso, Animesh Trivedi, Jan S. Rellermeyer, Lin Wang, Alexandru Uta, and Francesco Regazzoni. 2022. Future Computer Systems and Networking Research in the Netherlands: A Manifesto. *CoRR* abs/2206.03259 (2022). https://doi.org/10.48550/ARXIV.2206.03259 arXiv:2206.03259

[7] Guodong Jin, Xiyang Feng, Ziyi Chen, Chang Liu, and Semih Salihoglu. 2023. KÙZU Graph Database Management System. In *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org. https://www.cidrdb.org/cidr2023/papers/p48-jin.pdf

[8] Jean-Baptiste Lamy. 2017. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine* 80 (2017), 11–28.

[9] Chunhua Liao, Pei-Hung Lin, Gaurav Verma, Tristan Vanderbruggen, Murali Emani, Zifan Nan, and Xipeng Shen. 2021. HPC Ontology: Towards a Unified Ontology for Managing Training Datasets and AI Models for High-Performance Computing. In *IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments, MLHPC@SC 2021, St. Louis, MO, USA, November 15, 2021*. IEEE, 69–80. https://doi.org/10.1109/MLHPC54614.2021.00012

[10] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, Jianping Wu and Wendy Hall (Eds.). ACM, 270–288. https://doi.org/10.1145/3341302.3342080

[11] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin Van Eyk, Laurens Versluis, Vincent Van Beek, and Alexandru Iosup. 2021. OpenDC 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 455–464.

[12] Alessio Netti. 2022. *Holistic and Portable Operational Data Analytics on Production HPC Systems*. Ph. D. Dissertation. Technische Universität München.

[13] Natalya F Noy, Deborah L McGuinness, et al. 2001. Ontology development 101: A guide to creating your first ontology.

[14] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71. https://doi.org/10.1145/3434642

[15] Siddharth Samsi, Matthew L. Weiss, David Bestor, Baolin Li, Michael Jones, Albert Reuther, Daniel Edelman, William Arcand, Chansup Byun, John Holodnack, Matthew Hubbell, Jeremy Kepner, Anna Klein, Joseph McDonald, Adam Michaleas, Peter Michaleas, Lauren Milechin, Julia S. Mullen, Charles Yee, Benjamin Price, Andrew Prout, Antonio Rosa, Allan Vanterpool, Lindsey McEvoy, Anson Cheng, Devesh Tiwari, and Vijay Gadepally. 2021. The MIT Supercloud Dataset. In *2021 IEEE High Performance Extreme Computing Conference, HPEC 2021, Waltham, MA, USA, September 20-24, 2021*. IEEE, 1–8. https://doi.org/10.1109/HPEC49654.2021.9622850

[16] Axel Tenschert. 2016. Ontology matching in a distributed environment. (2016).

[17] Lamia Youseff, Maria Butrico, and Dilma Da Silva. 2008. Toward a unified ontology of cloud computing. In *2008 Grid Computing Environments Workshop*. IEEE, 1–10.

[18] Y Zhao, C Liao, and X Shen. 2017. *An Infrastructure for HPC Knowledge Sharing and Reuse*. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[19] Aolong Zhou, Kaijun Ren, Xiaoyong Li, Wen Zhang, and Xiaoli Ren. 2019. Building quick resource index list using wordnet and high-performance computing resource ontology towards efficient resource discovery. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 885–892.