A Data Center Simulation Framework Based on an Ontological Foundation

Ammar Memari[1], Jan Vornberger[1], Jorge Marx Gómez[2], Wolfgang Nebel[1]

Abstract

The IT-for-Green project aims at developing the next generation of Corporate Environmental Management Information Systems (CEMIS). Green IT being one important aspect of this, the IT-for-Green project seeks to support the analysis of a given data center situation and to support the simulation of alternative architectures to find ways to increase energy efficiency. To facilitate this, we develop a data center simulation framework, designed to be part of a larger CEMIS platform. This is achieved through a focus on flexibility, high interoperability and open standards. Flexibility is especially achieved by building the framework on top of an underlying data center ontology. This ontological approach allows us to derive many components of the framework from a single source, maintaining the ability to quickly adapt to future requirements.

1. Introduction

Spending on data center infrastructure is projected to surpass $126.2 billion in 2015 [1], and in line with this, energy demand of ICT infrastructure is rising continuously and almost doubling between 2000 and 2006 [2], [3]. While server hardware has generally become more energy-efficient, it has increased even more in performance, resulting in a net increase in energy consumption. This has reached a point where energy costs are now a dominant factor in the total cost of ownership [4].

This development has spurred new research into more energy-efficient data center architectures. One approach in this research is to build some type of model of the data center, which can then be used to quickly iterate on the design and try to optimize both individual components (depending on how fine-grained the model is) as well as the interplay between them. But besides optimizing the architecture, such a model can also be used to get a better understanding of which application workloads are using the most resources. Creating accountability in this way can create a feedback loop that helps to drive efficiency improvements on the software side.

The research we describe in this paper was conducted as part of the IT-for-Green project. This project aims at developing the next generation of Corporate Environmental Management Information Systems (CEMIS) and also addresses aspects of Green IT in its context. For those businesses which operate their own data center, the IT-for-Green project develops tools to help with analyzing their current data center situation and allow simulating the effects of changes and upgrades on the infrastructure to be able to find ways to increase energy efficiency.

In the remainder of this paper we describe a simulation framework that we have designed to meet these goals. The following section will explain the approach we took. Section 3 will then detail the complete simulation framework. We tevaluate the framework in section 4 based on the initial goals.

2. Materials and methods

Since our framework is part of a larger CEMIS solution, one of our main design requirements is flexibility and high interoperability. We have therefore taken great care in choosing appropriate

[1] University of Oldenburg, 26111 Oldenburg, Germany, {ammar.memari, jan.vornberger, wolfgang.nebel}@uni-oldenburg.de, Department of Computing Science, Embedded Hardware Software Systems

[2] University of Oldenburg, 26111 Oldenburg, Germany, jorge.marx.gomez@uni-oldenburg.de, Department of Computing Science, Business Informatics I / Very Large Business Applications

open standards where applicable and a flexible architecture to facilitate integration with other systems. This also led us to the decision of utilizing an ontology as the foundation of the framework. The main goal here is to have as few hardcoded components in the framework as possible, and derive everything from a single source instead. Evolving the framework is then a matter of updating this source and having the updates propagate through the rest of the system.

An ontology is a capable tool for reaching this goal. We can capture all the components of a data center and their possible interactions in the form of a rich ontology. From this we can then derive many different tools and other representations needed for a complete data center simulation. By choosing a standard ontology format (OWL), we also take our goal of compatibility and interoperability with existing tools and systems into account. Among other things, we derive a toolbox for a graphical editor from such an ontology. This editor can then be used to model an existing data center to prepare for a simulation. One goal here is to provide a reasonably fast and user-friendly way of creating such a model, also taking into account that the user may want to import inventory data from other sources.

Finally, we have decided to target the modeling language Modelica to be used in performing the actual simulation. It is an object-oriented, declarative, and multi-domain modeling language, and by leveraging it we can reuse many existing modeling tools. The next sections will describe each of these steps in more detail and how they tie together to form the full simulation framework.

3. Simulation framework

As the previous section already touched on, the workflow of the simulation framework looks as follows: Domain experts create the foundational ontology (we provide an initial attempt at that, but it can be extended further). From that ontology a toolbox for the graphical data center editor is generated. The editor is used by a user of the framework to create a model of a data center he or she wishes to study. This step is supported by both syntactic as well as semantic checks of the model, enabled by the underlying ontology. This model is then converted into a Modelica output using a Modelica component library as an additional input, and then used to run the actual simulations.

3.1. Data center ontology

In preparation for designing an ontology for the data center, we studied a number of ontologies which might be relevant in this context. The following is just a brief summary of our results, more details can be found in [5]. We looked at ontologies from three main categories: Top-level ontologies are ones that provide definitions for general-purpose terms, and act as a foundation for more specific domain ontologies [6]. Mid-level ones come directly underneath and are a bit more domain-specific. Low-level ontologies are the domain-specific ones defining terms and relations of a certain narrow domain. For example, [7] have built their low-level “Green IT” domain ontology complying with the mid-level “observation and measurement” ontology of [8], which in turn is built in conformance to the DOLCE foundational ontology [9].

Authors in [9] characterize foundational top-level ontologies to be ones that (1) have a large scope, (2) can be highly reusable in different modeling scenarios, (3) are philosophically and conceptually well-founded, and (4) are semantically transparent and richly axiomatized. Ontologies belonging under the same top-level ontology can be easily integrated; therefore we find it vital not to start from scratch by defining our own top-level terms but to focus on our domain and make sure our work integrates well by keeping its compliance to well-known top-level ontologies.

We selected DOLCE as our top-level ontology to build upon. It is a fairly abstract ontology dealing with basic concepts from a philosophical point of view. It defines four top level categories [9]:

- Endurants are entities which are wholly present at different times. In our case these are things like servers, routers or racks.
- Perdurants are entities that are extended in time and therefore have different parts at different times. The categorization of an entity as endurant or perdurant is often a matter of the desired temporal resolution, as most endurants become perdurants over long enough time spans. In our case perdurants are things like a power failure or a load balancing procedure.
- Abstracts are entities outside time and space, like the number “26”.
- Qualities map particulars to abstracts, such as the temperature on a specific point of a hot aisle is assigned to “26”.

An alternative top-level ontology would be SUMO [6] and to get the best of both worlds, we define mappings to a number of SUMO's concepts. For example:

```
GreenIT#Server           ===    SUMO#Server
GreenIT#Cooling        <-is-a-  SUMO#AirConditioner
GreenIT#ACPowerSource    ===    SUMO#ACPowerSource
GreenIT#Rack           -is-a->  SUMO#ChestOrCabinet
```

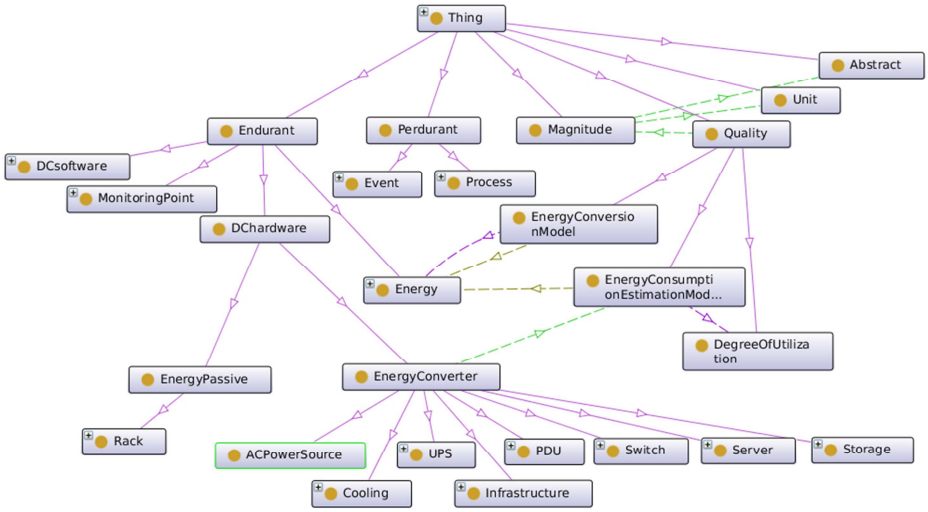

Figure 1: Data center ontology

On top of this base layer, important aspects we specify are three abstract flows in the data center: Power flow, heat flow and network traffic flow. All data center elements are looked at from the point of view of their effect on these flows. Therefore elements of the data center are viewed as `HeatSources`, `HeatSinks`, `EnergySources`, `EnergySinks`, and/or `TrafficFilters`. For example, a server is a `HeatSource`, an `EnergySink`, and a `TrafficFilter` whereas an air conditioner is a `HeatSink`, and an `EnergySink`. Figure 1 shows the current state of the ontology as a work in progress that is nevertheless usable for our requirements, as the rest of the simulation framework is derived from it.

3.2. Toolbox generation and document template

Based on this ontology, we then generate a toolbox in the next step, which can be used to model a specific data center. The generation of the toolbox is performed with the help of an XSLT file. The goal here is to provide building blocks, which can be assembled in a graphic interface to build a model of the data center. We chose to extend the graphical editor VUE (Visual Understanding Environment, [10]), which is provided with the generated toolbox and enables the user to build the

data center model by dragging and dropping the provided components onto a canvas and specifying relevant relations (e.g. heat exchange, electricity exchange). The user does not necessarily have to start from scratch for this task, as it is also possible to import CSV files of data center inventory, which can be used as a starting point to quickly model an existing data center.

3.3. Design verification

As the toolbox that was used to build the data center model is based on our initial ontology, we can leverage the ontology again to run a number of consistency checks in the next step. These are both syntactic checks (e.g. every server needs to have a label) as well as more complicated semantic checks. An example for the latter would be the requirements, specified in the ontology, that every server needs to be at the receiving end of a `sendsPowerTo` relation. A semantic reasoner, we chose HermiT [11], is used to verify all these requirements. Any errors that are found are transformed into a visual representation, and presented back to the user for correction.

3.4. Operational simulation model

After ending up with an error-free data center design, the workflow goes further into generating an operational model out of the design using again an XSLT transformation. This model is written in Modelica language. To follow best-practices and keep intra-component mathematical models apart from inter-component relations, the model is generated as two linked parts: Modelica package as a library that contains components' mathematical models, and the operational data center design which contains relations between instances of these components, and refers to component models from the former.

Data center Modelica package:

This package contains all component models and serves as a library. It is generated from the metamodel ontology through an XSLT transformation. Object-oriented quality of Modelica is utilized in this package to define generic types of component models, and then parameterize them with training data into final models of concrete components. An example of a generic server model is shown in figure 3. The code defines a server model that relies on a lookup table of load and consumption to predict by regression consumption value for a given load. The lookup table set in this generic model would be assigned values later in the concrete components.

```
block ServerLookup
  Modelica.Blocks.Interfaces.RealInput LoadIn
  Modelica.Blocks.Interfaces.RealOutput EnergyConsumption
  Modelica.Blocks.Tables.CombiTable1Ds lookupTable(table = fill(0.0, 0, 2))
equation
  connect(lookupTable.y[1],EnergyConsumption)
  connect(LoadIn,lookupTable.u)
end ServerLookup;
```

Figure 2: An excerpt from the DC package showing the definition of a lookup model of a server

```
block ProLiant_DL360
  extends ServerLookup(lookupTable.table = [0.0,0.001;1.0,35.0;20.0,35.0;50.0,60.0;100.0,90.0]);
end ProLiant_DL360;

```

Figure 3: An excerpt from the DC package showing the model of a concrete server as an extension of a generic model

Another generic model of a server could be formulated by measuring its consumption at the idle point where zero load is applied, and then again when it is 100% loaded. Under the assumption that between these points consumption relates to load in a linear fashion, we get a simple model of a server which requires minimal training but provides lower prognosis accuracy.

Concrete component models are defined as extensions of the generic models. Extension is performed in the general case by setting values to certain parameters. For example, the server

`HP_ProLiant_DL360` is modeled using the `ServerLookup` as shown in figure 4. Training data concerning the HP_ProLiant_DL360 is used for setting the lookup table values. Models of concrete components are instantiated in the data center design explained next.

Operational data center design:

This design is a translation of the data center XML design into the Modelica language. It captures, however, only functional properties of components and relations ignoring information about their physical location and layout. Instances of the components are related to their corresponding classes contained in the data center package where all intra-component information resides. Simulation tools that run Modelica models, such as the OpenModelica Shell, are able to run this design and return simulation results. The design can be run with live load measurements as input stream, outputting energy consumption and exhaust heat production at each component and measurement point. It can be run on historical or presumed load data as well with the aim of examining different what-if scenarios.

Hardware profiling:

In building the data center Modelica package we face the challenge, that data sheets for data center components are often not detailed enough to allow for the construction of an accurate simulation model of this component. To address this challenge, we develop tools to quickly profile a given component (i.e. measuring energy consumption in different load situations) and then perform regression analysis to derive an approximate model to be used in Modelica.

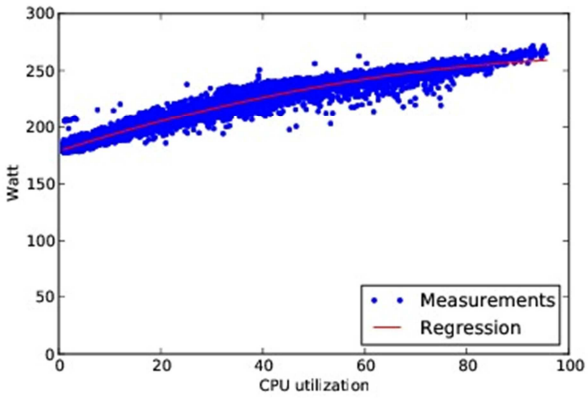

Figure 4: CPU activity vs active power for a test server

One major category of components we look at this way contains the servers themselves. They typically exhibit different energy consumption depending on their current work load. A number of researchers have looked at creating energy consumption models for servers [4], [12]–[19]. Some of the simpler models assume that the CPU is the most important factor and postulate a roughly linear relationship between CPU utilization and energy consumption. Authors of [12] have shown that even such a simple model can work fairly well and be within 10% mean accuracy. Of course this works best if indeed the CPU is the biggest consumer of energy. In combination with memory, the CPU did indeed dominate total power in the research done in [13]. An additional beneficial factor for a CPU-only model might be the fact that activities of the CPU seem to be well correlated with the power consumption of the rest of the system, as authors of [14] were able to show. Further successful applications of a simple CPU-based linear model can be seen in [13], [15]–[17].On the other hand, authors of [4] claim that the CPU no longer dominates platform power in modern servers and expect this trend to continue, as processors are adopting more and more energy-efficiency techniques. This is supported by the findings in [18], which reports that power

consumption of a server can vary by a factor of two while CPU utilization levels stay constant. Authors of [19] go as far as suggesting that accurate power modeling is only possible if the specific workload running on the server is taking into account and report errors as high as 50% in application-oblivious power models. Because of these different approaches to server models, we have striven to keep the ontology and simulation framework flexible enough to support different kinds of models. At the same time we have implemented a simple CPU-based linear model to be used as a starting point.

Our current approach to profiling servers and other load-sensitive components is therefore to instrumentalize the component temporarily with power measuring equipment and at the same time record the system status (for example CPU utilization) via SNMP. This workflow can fairly easily be carried out inside a production data center, as it requires no software agents on the servers and therefore poses minimal risk to the production work load. On the collected data (see figure 5 for an example) we run regression analysis to build models for the data center Modelica package. For a better accuracy of the models, we try to generate them based on further parameters besides the CPU utilization. The aim of this is to produce the model as the polynomial:

$$P = x_1 * C + x_2 * M + x_3 * I + x_4 * D + x_5$$

Coefficients vector (x) of this polynomial is estimated from the measured data by calculating multiple correlation. The method conducted by [20] is widely used in this regard, and is very well supported by the various tools. The approach used for estimating and testing the models starts by loading the measured data as a multi-column matrix containing power consumption (P), CPU utilization (C), used memory bytes (M), IP datagrams sent and received (I), and disk read and written bytes (D) measurements at specific timestamps. After loading and preparing the matrix, further variables are generated. These include the square of each measured variable so that correlation is calculated later for the squares and not only the linear variables. The next step is to apply the method of [20] to calculate the coefficients then compose the model. Calculated power consumption is then added to the matrix as an extra column, and the difference to the measured consumption is visualized.

4. Results and discussion

The sought-after flexibility of the framework is defined in terms of adaptivity to changes in DC hardware components and their interrelations. This attribute is measured by how smoothly these changes are incorporated and streamlined. To discuss this, we propose an example of an emerging technology in DC power supply and study the effect of its incorporation into the framework. The example technology is summarized in using Direct Current (DC) power in the data center instead of the Alternating Current (AC) power [21]. This is motivated by the energy savings that result from avoiding AC to DC and DC to AC conversions at the UPS, at the PDU, and at the front end of the power supply unit (PSU) on servers. Evaluating this trend is out of scope for this article, but we will focus instead on outcomes of its adoption, and ability of the framework to absorb them.

Researchers supporting this idea suggest that one AC to DC conversion takes place at the UPS. All power thereafter is DC power. This would require a new class of UPS that does not convert DC power coming out of its battery back to AC power, but rather delivers it as it is as DC power output. A new class of PDUs for distributing DC power is recommended, and a new PSU class which would result in a new DC server class is required. These new classes of equipment come accompanied with new restrictions on their power inputs and outputs.

Representing the new classes and relations within the framework requires modifying the ontology representing the metamodel. This can be achieved simply by adding an attribute to the `PowerSource` and `PowerSink` classes that determines if the input or output is an AC or a DC.

This attribute is then inherited automatically by all classes related to the aforementioned two through the `is-a` relation. This includes the `Server`, `UPS`, and `PDU` classes among others. Setting its default value to AC releases the designer from the burden of changing its value in existing components. Same attribute is added to the relation `sendsPowerTo`, and then its domain and range are restricted to be `PowerSources` and `PowerSinks` with a complying type of current. New component classes that use DC are then added normally into the ontology with one difference from conventional components that is the value of this attribute. Mathematical models of these new components are inserted as well into the ontology advocating thereby its role as the central metamodel. In a following step they will be automatically extracted from the ontology and collected into the data center Modelica package as mentioned earlier.

This is all it takes for the new components to be fully incorporated into the framework. Changes in the ontology are reflected automatically on the toolbox which will start listing the new components making them available for the designer. Semantic verification would check the domain and range of `sendsPowerTo` relations and detect inconsistencies without any additional effort. Generation of the operational model would translate the verified design including the new components into Modelica code and link them properly to their mathematical models. Flexibility is additionally incarnated in the ability of choosing different mathematical models to represent the same component. This allows seamless update of the component models through the ontology without having to edit the data center design itself.

We define framework interoperability as the ability of the framework to work in conjunction with other systems. This feature is achieved in our case mainly through relying on open standards like XML, OWL, SVG and XSLT which enables interoperability out of the box. Additionally, interoperability is approached by keeping the metamodel ontology aligned with widely-used top-level ontologies like DOLCE [9], and SUMO [6] as mentioned in section 3.1 and detailed in [5]. This alignment allows interoperation on the data level with other systems that utilize these top-level ontologies or their descendants, and the number of such systems is not to be underestimated. For example, ontology alignment allows our framework to seamlessly integrate data collected from monitored data centers around the globe through alignment with the ontology that is created in [7] to serve the purpose of such data curation. This data serve parameterize mathematical models of components with training data as mentioned in section 3.4 without having to go through the training phase. This interoperability on the data level is achieved by aligning the two ontologies, which is a relatively simple task having in mind that both correspond to DOLCE.

5. Conclusion

In this article we demonstrated a framework for data center modeling. The framework is composed of an ontology as a metamodel, and a workflow of transformations leading the designer through the design process from the toolbox into a fully operational model. All stages of the design throughout the workflow are governed by and accorded with the metamodel ontology, which was built with interoperability in mind. Additionally, we have included tools for hardware profiling to provide basic component consumption models. The framework was discussed later from the flexibility and interoperability points of view. The studies were based on actual example cases where these features exhibit a high demand. Having a central, interoperable, (re-) usable, and comprehensive metamodel proved useful under these circumstances and conferred flexibility and interoperability on the framework as a whole. Additionally, sticking to open standards in representing the metamodel, transformations, and the different stages of the design grants the framework additional interoperability, prevents a vendor lock-in, and allows a wide array of tools to access and manipulate data and processes.

6. Acknowledgement

This work is part of the project IT-for-Green (Next Generation CEMIS for Environmental, Energy and Resource Management). The IT-for-Green project is funded by the European regional development fund (grant number W/A III 80119242).

References

[1] Gartner, Inc., *Press Release: Gartner Says Worldwide Data Center Hardware Spending on Pace to Reach $99 Billion in 2011*. 2011.

[2] J. G. Koomey, *Estimating total power consumption by servers in the U.S. and the world*. 2007.

[3] EPA, US Environmental Protection Agency, "Report to congress on server and data center energy efficiency: Public law 109-431," 2008.

[4] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Comput.*, vol. 40, no. 12, pp. 33–37, 2007.

[5] A. Memari, "Angewandtes Semantisches Metamodell von Rechenzentren für Green IT," in *5. BUIS-Tage: IT-gestütztes Ressourcen- und Energiemanagement*, Oldenburg, Germany, 2013.

[6] I. Niles and A. Pease, "Towards a standard upper ontology," 2001, vol. 2001, pp. 2–9.

[7] C. Germain-Renaud, F. Furst, M. Jouvin, G. Kassel, J. Nauroy, and G. Philippon, "The Green Computing Observatory: A Data Curation Approach for Green IT," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2011, pp. 798–799.

[8] W. Kuhn, "A Functional Ontology of Observation and Measurement," in *Proceedings of the 3rd International Conference on GeoSpatial Semantics*, Berlin, Heidelberg, 2009, pp. 26–43.

[9] S. Borgo and C. Masolo, "Foundational Choices in DOLCE," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 361–381.

[10] A. Kumar and N. Schwertner, "Visual Understanding Environment," in *Third International Conference on Open Repositories*, 2008.

[11] R. Shearer, B. Motik, and I. Horrocks, "HermiT: A highly-efficient OWL reasoner," in *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008, pp. 26–27.

[12] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the 2008 conference on Power aware computing and systems*, San Diego, California, 2008, pp. 3–3.

[13] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 13–23, 2007.

[14] W. L. Bircher and L. K. John, "Complete system power estimation: A trickle-down approach based on performance events," in *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, 2007, pp. 158–168.

[15] S. Pelley, D. Meisner, T. F. Wenisch, and J. W. VanGilder, "Understanding and abstracting total data center power," in *Workshop on Energy-Efficient Design*, 2009.

[16] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunities," in *Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on*, 2005, pp. 275–280.

[17] M. Pedram and I. Hwang, "Power and performance modeling in a virtualized server system," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, 2010, pp. 520–526.

[18] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 807–812.

[19] R. Koller, A. Verma, and A. Neogi, "WattApp: an application aware power meter for shared data centers," in *Proceeding of the 7th international conference on Autonomic computing*, 2010, pp. 31–40.

[20] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. R. Soc. Lond.*, vol. 58, no. 347–352, pp. 240–242, 1895.

[21] C. Garling, *AC/DC Battle Returns to Rock Data-Center World*. 2011.