Proactive Workload Management for Bare Metal Deployment on Microservers

Daniel Schlitt[1], Christian Pieper[1] and Wolfgang Nebel[2]
[1]*OFFIS - Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany*
[2]*Carl von Ossietzky Universität Oldenburg, Ammerländer Heerstraße 114-118, 26111 Oldenburg, Germany*

Keywords: Bare Metal, Data Center, Energy Efficiency, Forecast, Microserver, OpenStack, Workload Management.

Abstract: This paper introduces a concept for an energy-aware workload management (WM) for heterogeneous microserver environments. Its main focus is on highly dynamic service-driven workloads often coupled to user requests requiring fast response times. The WM is developed in scope of the M2DC (Modular Microserver Data Center) project, in which a new server generation of composable microservers is designed. Targeting an easy industrial applicability, the underlying middleware is based on a turnkey OpenStack platform. As part of that middleware, the WM makes use of workload/utilization and power data as well as corresponding (prediction) models to deploy applications on the most suitable microservers and temporarily shut down unused capacities, either proactively or reactively (in case of deviations from forecasts). The WM has been implemented and simulated within a virtual environment. However, the integration, refinement and evaluation on the new M2DC hardware is still work in progress.

1 INTRODUCTION

Technically, the workload handled by computer systems can be separated into two rough types. First, there are high performance workloads or batch jobs which are placed once and just need to be completed in a certain time frame. Besides, there are transactional or service-driven workloads, which usually consist of smaller requests placed by users expecting a timely response. In terms of workload management (WM), the first mentioned domain benefits from popular management tools like SLURM, while the latter have either limited support or functionality – especially in commercial off-the-shelf (COTS) management SW.

Nonetheless, dynamic management of transactional workloads holds high potentials. The tight user coupling makes a both effective and efficient static operation complicated, as the workload and user behavior may be highly dynamic. That means the hardware capacity has to be specified for the maximum user demand based on experience and estimations. Additionally, low response times are highly important from the user's point of view. These factors make it impossible to find an optimal static allocation, as the dynamic workload behavior has to be considered for compute node selection and application deployment.

In contrast to high performance data centers, which try to fully utilize all of its compute nodes at any time, in the case of service-based computing the hardware usage has to be adapted to actual user demand. By dynamically adapting the number and selection of active nodes, the energy efficiency of data center operation can be massively increased.

Therefore, we realize an energy-aware WM based on a two-folded approach: (1) we optimize the current state of operation (active compute nodes, application allocation) based on workload/utilization predictions as well as performance/power models and (2) in the case of non-matching forecasts we use a reactive allocation to adjust the amount of deployed applications, while the deviating prediction models are re-trained. This currently ongoing work is done in scope of the EU H2020 project M2DC[1] (Modular Microserver Data Center), which aims to provide full server appliances with a convincing total cost of ownership (TCO). The project is based on three pillars (Oleksiak et al., 2016): It offers freely configurable heterogeneous microservers of different architectures (x86/64, ARM64) and hardware accelerators (GPU, FPGA). Furthermore, there is a layer of advanced management strategies e.g. to optimize the overall en-

[1]http://www.m2dc.eu

Schlitt, D., Pieper, C. and Nebel, W.
Proactive Workload Management for Bare Metal Deployment on Microservers.
DOI: 10.5220/0006762202460253
In *Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2018)*, pages 246-253
ISBN: 978-989-758-292-9

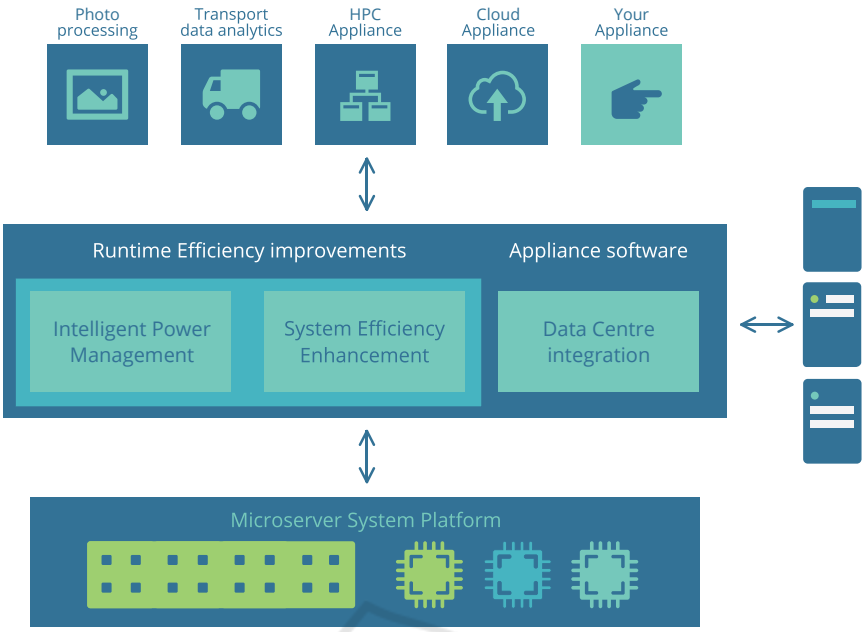

Figure 1: M2DC project scheme.

ergy efficiency. The described WM within this paper would be part of that layer. In order to enable an easy industrial applicability, M2DC is designed as turnkey platform with well-defined interfaces to the surrounding ecosystem (see Figure 1). For that purpose, the middleware is based on OpenStack, which is an open source software project offering a free cloud computing architecture. It is backed up by a large community, which steadily extends OpenStack's functionalities. New functions and features are made available through components, which implement e.g. support for networking infrastructure (Neutron), workload scheduling (Nova), management of bare metal resources (Ironic) or a web interface (Horizon). The WM, introduced in this paper, is part of the middleware and has the objective to dynamically optimize the server operation to contribute to the overall TCO reduction. As OpenStack including the Ironic[2] component – its bare metal provisioning module – is used as a base platform for the middleware, the WM is conceptionalized as an add-on to OpenStack.

In this paper, we describe the conception of the WM. Following the Introduction, Section 2 gives an overview on known approaches for WM in data centers. After that, we present our current development stage. The implementation is done in two separate places. On the one hand, we created an intelligent management component containing required estimation models and forecast algorithms. This component initiates the proactive and reactive node management and is able to access our OpenStack Nova[3] extension which was implemented on the other hand. As Nova is responsible for the actual node scheduling, we introduce new filters for an improved server selection based on energy efficiency and performance. Accordingly, Section 3 summarizes all relevant data sources and models used by the allocation algorithm. The general structure of the intelligent management including workload management, the allocation algorithm and the scheduling process will be explained in Section 4. Afterwards, Section 5 shows preliminary simulation results before Section 6 presents the conclusion and an outlook to the near future.

2 STATE OF THE ART

In research, there are dozens of approaches and implementations for server WM in data centers. Jennings and Stadler did an exhaustive literature survey and present their results in (Jennings and Stadler, 2015). We will focus on related work with respect to the most striking aspects of M2DC's WM, being (1) management of heterogeneous hardware nodes, (2) proactive management by using workload forecasts and server models and (3) setting up on OpenStack as most widespread cloud management system.

[2]https://docs.openstack.org/ironic/latest/

[3]https://docs.openstack.org/nova/latest/

Management of Heterogeneous Hardware. Jennings and Stadler (Jennings and Stadler, 2015) present an exhaustive list of WM approaches using dynamic virtualization. Some of them have also been applied to industrial products. A good example is the distributed resource scheduling (DRS) and its extension distributed power management (DPM) feature by VMware for its virtualization technology, offering significant energy savings by dynamic VM migration and minimization of physical machines (Gulati et al., 2012). Applying such solutions requires fully virtualized data centers, but virtualization technology is currently limited to x86/64 and ARM64 and it has some considerable overhead on small microserver nodes. Therefore, it is not the best solution for M2DC with its heterogeneous hardware philosophy, supporting a wide spectrum of different compute nodes ranging from low power to high performance. Still, there are some investigations using other technologies for WM. Piraghaj et al. (Piraghaj et al., 2015) as well as Kang et al. (Kang et al., 2017) propose WM based on Docker containers to additionally reduce the virtualization overhead, which would also be suitable for low-power microservers. De Assunçao et al. (de Assunçao et al., 2016) implement dynamic server provisioning using bare metal provisioning via OpenStack Nova and an extension of its built-in filter and weighing scheduler. However, these alternative approaches still do not include accelerators like FPGAs or GPUs.

Proactive Workload Management. Some approaches consider workload forecasts in order to achieve better consolidation rates with less buffer capacities. Zhang et al. (Zhang et al., 2014) propose an algorithm which schedules work in a hybrid cloud between on- and off-premises compute capacities. On-premises in terms of software or workload which is processed on company owned or rented servers, whereas off-premises means workload processing on remote facilities like cloud computing or Software as a Service (SaaS). The researcher divide workload in 'base' load and 'flash crowd' load, which are managed individually. Base load is managed proactively on-premises (90 % of time within 17 % prediction error) and flash crowd load is managed reactively off-premises. Currently, their approach does not allow for dynamic workload scheduling and it is restricted to certain workload types, which impedes an automatic usage on unknown applications. Herbst et al. (Herbst et al., 2014) use time series analysis to forecast workloads and use this information for proactive resource scheduling via virtualization. They present an exhaustive comparison of diverse statistical methods. The most suitable time series model is selected at run time based on the given context. By dynamically selecting the model, the relative error could be reduced by 37 % on average compared to statically selected fixed forecasting methods. A similar approach is utilized in M2DC.

Cloud Management based Realization. The adaptability in real, productive data centers is an important criterion for the success of novel WM methods. The best approach would be to set up on broadly known and accepted server/cloud management tools to have a potential user base with fewer contraints compared to proprietary approaches. There is only little work in research, which already use common management tools as a base for implementing their own solutions. The most promising one is from Beloglazov and Buyya (Beloglazov and Buyya, 2015), who use OpenStack for their dynamic consolidation module called Neat. It is designed as a transparent add-on not requiring modifications on OpenStack installations. Neat makes use of OpenStack's management functions via public APIs to perform the VM migrations planned by the Neat allocation algorithm. However, Neat is restricted to virtualization and therefore not the optimal solution for M2DC's smaller microservers. Fujitsu is also working on a management tool called FUJITSU Software ServerView Resource Orchestrator (ROR) (Yanagawa, 2015) building up on OpenStack. In contrast to Neat, autonomic functions for WM are missing, as the focus rather is on increasing the ease of operation to make management functions usable for business purposes. Then again it is planned to integrate OpenStack Ironic in the future to support bare metal deployment next to virtualization.

M2DC's energy-aware WM combines the advantages of several referenced approaches which each focus only on limited aspects. By applying a combined proactive and reactive allocation algorithm, the M2DC WM is able to optimize aggressively while also providing emergency measures for sudden spikes in workload. However, the most appealing advantage of M2DC's approach is the strict focus on future applicability. While the usage of OpenStack as base platform should help spreading the solution due to its broad community and popularity, the consideration of alternative compute nodes in the modeling and management process guarantees a future relevance when GPUs and FPGAs become more popular in general-purpose computing.

3 MODELS

The WM improves the overall application-to-node allocations. For the decision making it requires data sources of directly measured operational parameters as well as models, which make estimations based on collected data. This section briefly introduces required parameters, but does not go into details. It is rather an overview for further WM explanations.

Data Sources. Data sources are separated in five parts: The *workload* is the amount of work a compute node has to tackle. Workload, in the form of jobs, may either be placed once for direct processing or may come over time based on current demand or user requests. *Performance* is defined by the workload processed per time for a given compute node. That means performance is also application specific and can therefore only be measured for adapted applications. Benchmarks help to measure hardware performance for application workloads in order to compare capabilities of compute nodes. This must be done prior to productive operations. *Utilization* represents the usage of a compute node's subsystems by the applied workload and can be used as objectively determinable proxy for workload data, when access to application data is not available. The *power* will be measured for each compute node, e.g. to determine the energy efficiency. In conjunction with resource utilization data, power models can be used to estimate power demand for arbitrary node utilization. The thermal and resource management aggregates the thermal state of compute nodes in a *thermal metric*. For this purpose it scores available temperature sensors or temperature affecting components like fan speed into a ratio, which can be used for later placement decisions.

Models. Using the collected data sources, models are able to make calculations for defined time points, even in the future. The WM requires models for power, performance, energy efficiency and workload/utilization. *Power models* are able to calculate the power demand based on historic and future workload or utilization trends. Power data is required for other models like the energy efficiency. *Performance models* determine the performance of M2DC components. In general, the objective is to use published benchmark results in order to avoid measurements on productive systems, cf. (Schlitt and Nebel, 2016). As the server benchmarks cannot be applied to every hardware type of M2DC compute nodes, an application-specific method will be required. Therefore use case applications must be designed considering external monitoring of performance factors. The *energy efficiency model* is defined as relation between performance and the amount of required energy. *Workload modeling* is a main selling proposition of M2DC's WM. It utilizes workload or utilization forecasts for the scheduling process, taking into account the user behavior to enable proactive actions for regular peaks. For this, the WM uses a number of univariate, statistical procedures to automatically model unknown, generic time series solely based on historic values. These are Seasonal and Trend decomposition using Loess (STL), Holt-Winters (HW) filtering and Seasonal AutoRegressive Integrated moving Average (SARIMA), which results are compared to choose the best approach for the specific situation.

Storage. Measured values are stored in a central database using Gnocchi[4] of the underlying OpenStack, while models required for the estimation of values are placed within the intelligent management.

4 WORKLOAD MANAGEMENT

The WM component is embedded in the intelligent management (IM) layer of M2DC's middleware. The IM layer is depicted in Figure 2. The central component is the IM component, which acts as an interface combining dataflows and workflows between M2DC's management components, the (model) data and OpenStack.

The IM is the connection between the resource and thermal management (RTM), the Power Management (PM) and the WM components. There are two different PM solutions, one of them switching host states by using the OpenStack API and the other using directly the server controller API in case further power management functionality is needed (e.g. power capping) or if usage of OpenStack and WM is not desired. Basically, WM and RTM could have different constraints, which have to be negotiated before instructions are sent to the PM. Access to the server models (power, performance, ...), thermal and workload models as well as measured data in the Gnocchi database is realized through references to the components, gathered in the IM class. Thereby, RTM and WM can access each kind of model and use every information available for management decisions. Moreover, the IM represents the interface to OpenStack Nova regarding the dynamic scheduling. While the allocation algorithm itself is part of the WM class, connection to the Compute API as well as the filter

[4]http://gnocchi.xyz/

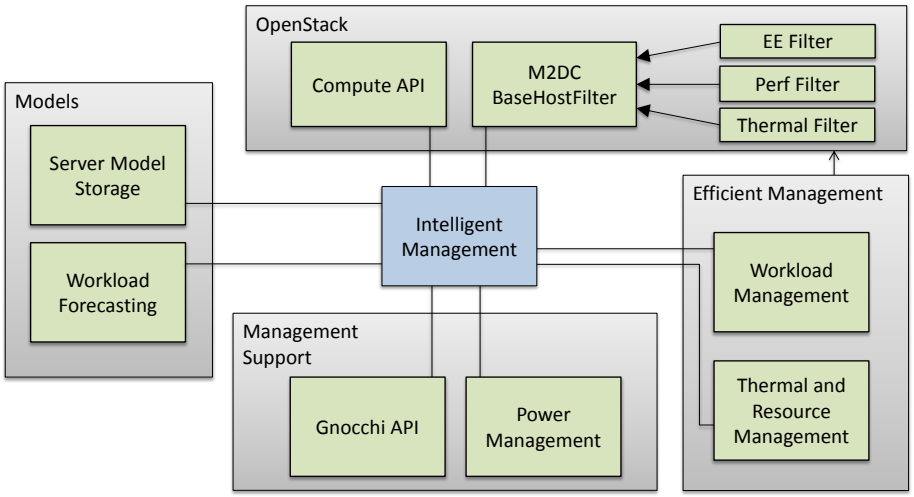

Figure 2: Overview on intelligent management component.

classes are established via IM. The Compute API is used for triggering application deployment and setting host power states.

The WM's main objective is to optimize the application-to-node allocation in order to increase the overall energy efficiency, i.e. the WM tries to select the most suitable compute nodes for the applications regarding current and future workload demands – e.g. using low power ARM nodes if utilization is low at night. Therefore, the current and future state is analyzed and corresponding actions are triggered by an allocation algorithm. Suitable actions are executed by the corresponding components (e.g. power management for host power actions), which are accessed via IM. Moreover, the WM provides general methods for creating, deploying, moving and destroying application instances, which are called, when the user takes the corresponding actions in OpenStack. Also, the WM has means to read the current state of hosts/compute nodes and instances inside OpenStack, which are analyzed by the allocation algorithm. These methods ensure that the allocation states between WM and OpenStack are consistent.

The allocation algorithm is the centerpiece of WM. Generally, it distinguishes between two operational states: proactive and reactive allocation. The operational state is constantly determined by comparing current, actually measured utilization/workload with forecast values. If the deviation is within a tolerance range (e.g. 20 %), allocation optimization is done proactively based on forecasts. If the actual utilization is below the threshold, the allocation will still be optimized proactively except that the workload modeling process for the affected compute nodes will be repeated beforehand (maximum once per day). In case the tolerance thresholds and the node utilization thresholds (e.g. 80 %) are exceeded, allocation optimization is interrupted and a high priority reactive allocation will be performed. Afterwards, workload models of concerning nodes will be reprocessed and the proactive allocation optimization continues.

In proactive operation, the allocation algorithm analyzes the current and future host utilizations/workloads and decides for each node (1) if additional compute capacities have to be provided on short-term, (2) the state may be optimized or (3) if it should remain constant. As booting and deployment activities take several minutes, a time window of at least 30 minutes should be chosen for workload analyses. Then again, the window must not be too long, as that would reduce the optimization potential because the allocation has to be valid (not exceeding utilization thresholds) for the whole time window.

The algorithm starts to examine the provisioning of more compute capacities. This is necessary, if the expected utilization of a single node will exceed its utilization threshold (e.g. 80 %) in the investigated time window. This will initiate the deployment of the affected application on another node, as every node is expected to maintain a certain predetermined buffer. The scheduling of a new node will be triggered and executed in time to ensure the necessary capacities are available as soon as they are needed. Subsequently, the original node will be powered down, if the new node has enough computing capacities. After provisioning additional capacities, the utilization trend is analyzed for the new allocation and the process is repeated, if necessary.

When all cases for capacity provisioning have been resolved, the algorithm looks into potential energy efficiency optimization, which is often achieved by utilization decreasing over time (e.g. after-work

hours, night). The algorithm starts with the currently least energy efficient node (determined by energy efficiency model) and looks (1) for potential consolidation of the running application instance or (2) for potential alternative nodes capable of running the same application more efficiently. Instance consolidation is possible, if all other nodes running the same application are able to provide enough capacities to take over the workload of the instance which will be shut down. Thus, an instance and node shutdown will be triggered and called in OpenStack Nova via IM and the Compute API. If this approach is not feasible, the algorithm checks the availability of a more energy efficient node for given utilization. If energy would be saved in the investigated interval despite deployment overhead and buffer utilization is still met, the scheduling as well as a subsequent shutdown is triggered for the selected node/application. Anyway, the next node is examined in the same way until every node was examined. The proactive allocation process will be restarted when half of the investigated time window is over, thereby ensuring that future utilization increases can be handled in time.

A reactive allocation is done for each compute node, whose actual utilization exceeds its forecast by the predetermined threshold. Thereby, the scheduling of an additional compute node for the affected application will be triggered before doing anything else, as one can no longer rely on the utilization forecasts and it is not known how the current trend will continue. Moreover, the scheduling will be configured to pick preferably high performance nodes in order to countervail a further increasing utilization. After all necessary reactive operations have been performed and workload modeling has been rerun, the allocation algorithm tries to proactively optimize the current state.

All mentioned configuration parameters (thresholds, percentages, times) are preliminary and have to be adapted based on empirical evaluations.

4.1 Scheduling via Nova

The actual scheduling of applications to compute nodes is done via the OpenStack Nova FilterScheduler[5], using several host filters and weighers. This scheduler is extended by an `M2DCBaseHostFilter`, which is derived from the Nova `BaseHostFilter`. It has three child classes representing the considered aspects in WM and RTM. With regard to WM there is an energy efficiency and a performance filter. The energy efficiency filter passes through only nodes/hosts that have at least the minimum specified energy efficiency threshold, which was assigned by WM. The utilization dependent node energy efficiency is computed by using the models suggested in Section 3, which are accessed via IM. The performance filter works analogous. Regarding RTM, there is a thermal filter passing through nodes/hosts based on a thermal metric, which indicates if the corresponding node should be considered for deployment or if the thermal environment is currently not suitable (e.g. thermal hot spots). After the filtering process, the most suitable host is selected by weighed sorting of all passed compute nodes by using adapted `M2DCBaseHostWeigher`. The weights are again given by the WM and depend generally on expected utilization/workload levels, e.g. if the expected utilization is low, the more energy efficient nodes are preferred and would benefit from a higher weighing.

As the allocation algorithm examines the operational state of compute nodes and instances, it triggers scheduling processes for a given application with state-specific parameters defining energy efficiency, performance and thermal requirements. In case there is no suitable host for the triggered scheduling process (i.e. no host passed all filters), the allocation algorithm will get an appropriate response and it will ignore this node as optimization target until the next power management operation was resolved. If the scheduling was triggered due to reactive allocation, the administrators will be notified that there is no suitable host to provide the needed capacity.

4.2 Microserver Support

M2DC's main unique selling point is the support of a high variety of different hardware architectures in the microserver format. Within M2DC, OpenStack is extended to support dynamic composition and provisioning of diverse microserver nodes including FPGAs and GPUs. As WM is part of the base M2DC installment, all available microserver types have to be supported. This mainly influences whether an application is deployable on a certain microserver or not, depending on available implementations and (boot) images.

The allocation algorithm itself does not consider whether an application can be deployed on a certain microserver. This aspect is checked in the destination host selection of OpenStack Nova filter scheduler via the `ImagePropertiesFilter`. This filter only passes hosts satisfying the requirements given with the application image to deploy. If multiple images (e.g. for different architectures) are available, several requests will be sent consecutively.

[5]https://docs.openstack.org/nova/latest/user/filter-scheduler.html

Table 1: First simulation results for static vs. dynamic (optimized) allocation.

Scenario	Configuration	Energy (static)	Energy (optimized)	Savings
1	2 x86 (2013) + 2 ARM64 (2015/16)	3372 Wh	2082 Wh	38.2%
2	4 x86 (2013)	3234 Wh	2382 Wh	26.2%
3	2 x86 (2017)	1710 Wh	1710 Wh	0.0%
4	4 ARM64 (2016)	1860 Wh	1440 Wh	22.5%

4.3 Power Management

The power management is an essential part of M2DC's IM in order to control the overall energy demand of the system. Realized as support component, it encapsulates access to different power actions, which comprise changes to power states but also functions like power capping or frequency scaling. M2DC's power management components use two different approaches. On the one hand, OpenStack needs to control nodes for its scheduling process, where e.g. hosts need to be turned on prior to deploying application instances. For the M2DC server, extensions will take care to implement host-actions functionality of Nova, respectively node-set-power-states of Ironic, using the provided M2DC server API. On the other hand, it is necessary to execute basic PM functions without using the OpenStack platform. Therefore, there is an additional power management implementation that accesses the M2DC hardware directly. These functions include fan management, power capping or frequency scaling demanded by RTM, which are accessed via M2DC server API or directly on the microserver.

5 PRELIMINARY RESULTS

So far, the IM component, the WM algorithms and the OpenStack Nova extensions have been implemented in a virtual environment based on DevStack. To test the functional capability as well as the effectiveness of the algorithms, we expanded the DevStack environment with a simulation. As simulation data we modeled several server systems (x86 servers from 2013 and 2017) and microservers (ARM64 from 2015 and 2016) with regard to performance, power and energy efficiency – each dependent on a given utilization. The models base to some extent on own measurements but for the most part on publicly available benchmark results such as SPECpower_ssj2008[6] and are defined by polynomial functions with utilization as variable. As input data we used utilization forecasts for two real application workloads measured in a productive data center. In our simulation, both applications may be run in several instances distributed over different compute nodes, but not more than one application instance per node (bare metal approach). We simulated several small node configurations each with a static allocation of application instances to compute nodes as well as a dynamic workload management. The simulation was accelerated by factor 60 so that the run-time of 10 minutes represented 600 minutes in reality. The results for running each configuration can be found in Table 1.

In scenario 3, there was no optimization feasible because the two applications have to run on different server nodes due to the bare metal approach (opposed to virtualization, which can potentially operate both applications on a single node). However, scenarios 1, 2 and 4 show significant savings even in small configurations with few optimization possibilities. It is interesting that the mixed configuration in scenario 1 is more efficient than the homogeneous configuration in scenario 2, if workload management is applied, while it needs more energy than scenario 2 in a static operation. This is an indicator for efficiency gain potential by using heterogeneous computing nodes. Another interesting result is the comparison of scenarios 3 and 4. While the modern x86 systems are better in a static operation, the ARM64 configuration provides a more fine-granular optimization potential, although this is somewhat impaired by the small scenario configurations.

We also simulated the productive server environment of one of M2DC's partners as of the year 2013 (which is the base year for comparisons in M2DC project) consisting of 40 x86 server systems. Therefore, we modeled the server systems from 2013 regarding performance, power and energy efficiency. We also measured the workload of our target application in 2016 and used the corresponding forecast model as input for the simulation. The result was that the application of WM could reduce the energy demand by about 800 kWh per month ($> 40\,\%$) by simply taking advantage of the high daily variance in workload profiles.

[6]https://www.spec.org/power_ssj2008/

6 CONCLUSIONS

The M2DC project consortium is working on bringing alternative server technologies like ARM based processors, GPUs and FPGAs into commercial data centers. The aim is to decrease capital and operational expenses by using microservers, which are both cost- and energy-efficient as they can be specifically selected and configured for given tasks. As partner in M2DC, we are specifically working on WM mechanisms which make use of workload/utilization, performance and power data to place applications on the most suitable microservers and temporarily shut down unused capacities. A first simulation of the algorithms show promising results of about 40% energy savings. The related work section shows that there is already much work in this field. However, the M2DC approach is the only one (to best of our knowledge) which combines the advantages of using workload/utilization forecasts, setting up on a commonly known platform (OpenStack) and supporting alternative (modern) server architectures.

The next steps on the M2DC WM include the automatic modeling of server power, performance and energy efficiency models as well as automatic model selection and modeling of application workload/utilization forecasts. After fine tuning and further simulations in the virtual environment the IM including its interface extensions to OpenStack will be integrated in the M2DC testbed containing the newly developed microserver nodes. Evaluations on this testbed with use cases defined in M2DC will then be published in a future research paper.

ACKNOWLEDGEMENTS

This scientific work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 688201 (M2DC).

REFERENCES

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. *Concurrency and Computation: Practice and Experience*, 27(5):1310–1333.

de Assunçao, M. D., Lefevre, L., and Rossigneux, F. (2016). On the impact of advance reservations for energy-aware provisioning of bare-metal cloud resources. In *CNSM 2016*.

Gulati, A., Holler, A., Ji, M., Shanmuganathan, G., Waldspurger, C., and Zhu, X. (2012). Vmware distributed resource management: Design, implementation, and lessons learned. *VMware Technical Journal*, 1(1):45–64.

Herbst, N. R., Huber, N., Kounev, S., and Amrehn, E. (2014). Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and computation: practice and experience*, 26(12):2053–2078.

Jennings, B. and Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23(3):567–619.

Kang, D.-K., Choi, G.-B., Kim, S.-H., Hwang, I.-S., and Youn, C.-H. (2017). Workload-aware resource management for energy efficient heterogeneous docker containers.

Oleksiak, A., Rosinger, S., Schlitt, D., Pieper, C., et al. (2016). Data centres for iot applications: The m2dc approach (invited paper). In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 293–299.

Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., and Buyya, R. (2015). A framework and algorithm for energy efficient container consolidation in cloud data centers. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pages 368–375. IEEE.

Schlitt, D. and Nebel, W. (2016). Data center performance model for evaluating load dependent energy efficiency. In *Proceedings on 4th International Conference ICT for Sustainability 2016 (ICT4S 2016)*. Atlantis Press.

Yanagawa, T. (2015). OpenStack-based Next-generation Cloud Resource Management. *Fujitsu Sci. Tech. J*, 51(2):62–65.

Zhang, H., Jiang, G., Yoshihira, K., and Chen, H. (2014). Proactive workload management in hybrid cloud computing. *IEEE Transactions on Network and Service Management*, 1(11):90–100.