International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 5, Number 2, pages 304–328, July 2015

Overhead-aware Load Distribution and System Shutdown for Energy-Efficient Computing

Jörg Lenhardt, Wolfram Schiffmann
FernUniversity in Hagen, Computer Architecture Group
Universitätsstraße 1, 58097 Hagen, Germany

Received: February 15, 2015
Revised: May 5, 2015
Accepted: June 1, 2015
Communicated by Susumu Matsumae

Abstract

The energy consumption of server farms is steadily increasing. This is mainly due to an increasing number of servers which are often underutilized most of the time. In this paper we discuss various strategies to improve the energy efficiency of a datacenter measured by the average number of operations executed per Joule. We assume a collection of heterogeneous server nodes that are characterized by their SPECpower-benchmarks. If a time-variable divisible (work)load should to be executed on such a datacenter the energy efficiency can be improved by a smart decomposition of this load into appropriate chunks. In the paper we discuss a sophisticated load distribution strategy and extend it by an adaptive power management for dynamically switching underutilized servers to performance states with lower energy consumption. Of course, also transitions to higher performance/energy states are possible if required by the current load. We introduce a new time slice model that allows a reduction of the switching overhead by means of a few merge and adjust cycles. The resulting ALD+ strategy was evaluated in a webserver environment with real Wikipedia traces. It achieved significant reductions of the energy consumption by the combination of load distribution and server switching by means of the time slice model. Moreover, ALD+ can be easily integrated into any parallel webserver setup.

Keywords: power efficient load distribution; power off; energy consumption; overhead;

1 Introduction

Modern data centers consume large amounts of electrical energy by computing devices as well as by cooling equipment. In the lifetime of a server system it causes energy costs that overtake their initial price by a large margin. In addition, this energy requirement also has an environmental impact. This results from increased emissions of carbon dioxide, as a large part of the electricity is generated by fossil fuels. Therefore, it is reasonable from both an economic and an environmental point of view, to reduce power consumption to the minimum that is necessary to provide the desired service. Two opportunities to achieve that goal in large computing environments are energy efficient load distribution and suspending/powering off under-utilized or unused systems.

In previous studies [10] we presented several load distribution strategies for power efficient load distribution. The algorithms generate load distribution schemes for various load levels of server

farms. The load is specified in a system independent measure as operations per second for the maximum and current overall load as well as the maximum and current load of a single server system. This measure is based on SPECpower_ssj2008 benchmark which provides performance and power data for a lot of hardware configurations since late 2006. SPECpower relies on Server Side Java (SSJ) for measuring power consumption of servers at various load levels [17].

Environments such as web servers are particularly suited for balancing the web requests in a power efficient manner using load distribution schemes. However, environments with predictable workload and loads in a certain granularity can also benefit from these distribution schemes: In a virtualized server environment virtual machines can be placed in a power efficient manner using load distribution schemes when having information about the future load.

This paper deals with the extension of the above mentioned load distribution algorithms. The idle power consumption of servers is usually between 20 % and 50 % of the maximum power consumption, in some cases even as high as 78 %. Our most efficient algorithms generate distribution schemes in which at lower utilizations more servers are unused or under-utilized. There is a great potential to reduce power consumption by suspending or powering off these systems. However, the overhead (time and energy) for switching a server's state (shutdown and reboot or suspend to and resume from main memory) has to be considered in this context as well. The decision to which state a server should switch in a certain time interval depends heavily on the length of the interval as well as power and energy consumption of the system.

A more formal description of the problem is defined as follows: Suppose a server farm consisting of n servers S_k with $0 < k \leq n$. Each server has a maximum load capacity of o_k^{max} and a current load of o_k, both measured in operations per seconds (ops). Above that, for each server a power function p_k is available which computes the power consumption in watts for a current load o_k. An overall load o shall be distributed so that the total power consumption of the server farm is minimized. To achieve that goal, the servers can be loaded differently, suspended or powered off. In all cases, no performance degradation is allowed while reducing the power consumption.

We examined at which point a server can be assumed to be under-utilized in dependence to the current load of the server farm and the utilization of the other servers in this environment. An extension to one of our previously proposed algorithms has been implemented which generates distribution and switching schemes for various load levels. In addition, we investigated the influence on the power efficiency at which the remaining running systems operate when applying the extended algorithm.

The remainder of this paper is organized as follows: Related work is presented in Sec. 2. Section 3 gives an overview over power efficient load distribution. In Sec. 4 we present our power minimization algorithms including the consideration of overheads for suspending or powering off nodes. Experimental results are presented in Sec. 5. We conclude in Sec. 6.

2 Related Work

Two approaches to construct a distributed system with regard to an optimized overall power consumption are low-power-computing (LPC) and energy-aware-computing (EAC) [5].

LPC uses many low power processors simultaneously. These can be tightly packed because their heat dissipation is very low. To enforce the computational power, parallel software is necessary. Compared to sequential programs, parallel versions are commonly harder to develop and are more expensive. LPC is mainly used in special architectures.

In contrast, EAC does not use special hardware but rather already implemented features of the traditional components such as processors, memory, network, etc. In addition, it has to be differentiated between node and system level. Technologies on node level are dynamic voltage scaling, turning off processor cores and request batching. Technologies on system level are coordinated voltage scaling, turning off nodes and virtualization.

Valentini et al. [18] present the state of the art of the above mentioned technologies and examine them regarding energy savings and performance slumps. Energy saving potential varies significantly in relation to application, workload, cluster system, and scheduling strategy. The future of dynamic

power management in clusters will include a combination of power-scalable components, each with its own management scheme. They conclude that LPC is currently too expensive.

Data centers for providing web services usually consist of a great variety of server types with different power over performance characteristics [1, 9, 12]. This heterogenity requires a smart distribution of the incoming service requests to the available machines. An overview of literature related to energy in web server systems is given in [16]. Due to the growing number of web users the energy costs and the negative environmental impact increase steadily and thus it will become important to not solely provide a high quality of service (QoS) but also to minimize the energy consumption [4, 3].

Fortunately, the web service request rate (and the associated load) varies in the course of time [11]. Thus, instead of running all the servers with maximum performance chunks of work can distributed to a farm of parallel web servers. The performance and power consumption of the servers will then be adjusted according to their current load. If a lower performance is requested the server's power management system can decrease the CPU cores' voltages and frequencies or it can also switch to various power states [18]. Unfortunately, the lower the power consumption of a power state the greater the latency for switching back to a higher performance values [15].

The major purpose of this work is to save energy by deactivating inefficiently operating nodes. An algorithm for a dynamic load distribution is presented in [3]. Based on the daily login rate on Windows Live Messenger, a predictive model of the desired load can be developed. On this basis, the server demand could be determined. Overheads through switching servers on and off are not considered.

The work of Guenter et al. [7] presents an algorithm for a dynamic load distribution based on a load prediction. Additionally, they consider overheads caused by the change from one power state to another. The power states are on, off, hibernation and stand-by.

3 Power Efficient Load Distribution

In this section we present an overview over power and energy benchmarks and load distribution algorithms. Subsection 3.1 deals with power and energy benchmarks in general and with the SPECpower_ssj2008 benchmark in particular. Various load distribution algorithms are briefly presented in Subsect. 3.3.

3.1 Power and Energy Benchmarks

Poess et. al. analyzed SPECpower and other energy benchmarks [13], among them Transaction Processing Council (TPC) and Storage Performance Council (SPC) benchmarks.

TPC-C and TPC-E benchmarks, provided by the TPC since 1988, measure On Line Transaction Processing systems (OTLP). TPC targets multi-tier systems built from several computers and interconnection hardware. Part of configurations are database servers, middle-tier systems and storage subsystems as well as connectivity devices like switches or routers. TPC benchmarks deliver objective and verifiable performance data. In 2007 TPC-Energy added energy metrics that co-exist with already existing metrics and allow power analyses of all components. Before that, power estimation of TPC-C and TPC-H was possible.

The SPC defines and standardizes benchmarks targeting storage subsystems consisting of storage units, adapters, controllers, and storage area networks since 1997. Results are objective, verifiable, and vendor-neutral performance data. Instead of building a new benchmark addressing the energy consumption of storage systems, SPC developed optional energy extensions (SPC-1C/E).

Compared to TPC and SPC, SPECpower was developed for power-to-performance analysis only. It targets a single server or a combination of identical servers. SPEC published over 460 benchmark results of various servers. As our interest is the power-to-performance ratio of single servers, the SPEC benchmark fits well for our purposes. The number of available results offers an adequate base for our analyses.

Throughout this work we are using results of the SPECpower benchmark. It relies on Server Side Java (SSJ) for measuring power consumption of servers at various load levels running Java applica-

tions [17]. Current servers include technologies which autonomously reduce its power consumption if the system is not fully utilized. Many of these technologies are recognized by the SPECpower benchmark. The benchmark itself runs workloads representing a server application used by a large number of users. Details on the target workload can be found in [17]. Power consumption is measured at various discrete levels of system load. There are two phases in an SSJ run: First, there is a calibration phase followed by a phase of real benchmark runs at a series of target loads. In the calibration phase the maximum throughput of the server is determined. Then, the actual benchmark runs take place. Each run performs some percentage of the calibrated throughput. These runs decrease target load from 100 % to 0 % in 10 % steps. Running at 0 % load is referred to as *active idle*: the application can accept requests, but none are actually received.

To evaluate the power-efficiency of a system, we examined the $ssj\_ops/watt$. Older hardware systems usually are most efficiently operated at 100 % utilization. This is valid for most newer systems as well. But we observed an increasing portion of systems which are most efficiently at loads from 65 % upwards.

3.2 Power Curve Fitting

Information is needed that relates load levels of a server to its power consumption. This information is already available for hundreds of popular servers and for not yet benchmarked systems it could be retrieved by the SPECpower benchmark.

As this information is represented as discrete values, a fitting algorithm calculates a cubic approximation of a power function p_k for each server S_k. These power functions are then used to calculate load partitions over a set of servers.

The power consumption of a server is generally simplified and modeled by a linear function. The literature [2] suggests linear interpolation between two points. The work of Hsu and Poole [8] shows that this simple model is appropriate for servers which were produced before August 2009. A maximum model error of up to 45 % occurred for newer servers. In order to reduce the maximum model deviation to an acceptable level, we use a cubic function [19]. Fig. 1 shows the difference between a cubic model and a linear interpolation. The cubic functions approximates the data points adequately whereas the linear function strongly deviates over a wide range from 30 % to 80 % overall load.

The cubic model leads to a maximum error below 10 % in almost all cases, the average error is below 5 %. This value is within the 10 % margin of error where a model has an acceptable accuracy [14]. Above that, as voltage scaling has quadratic and frequency scaling linear influence on the power consumption cubic fitting models reality accurately.

3.3 Load Distribution Algorithms

In previous papers we present various approaches to generate load balancing schemes to reduce power consumption of heterogeneous server farms [10]. Starting with a given workload in operations per second to be processed the work is distributed among a server farm to minimize power consumption. A function p_k models the power consumption of server S_k at a specific load given in operations per second.

Relative Load Balancing (RLB) distributes a workload so that all servers are utilized equally. In contrast, Absolute Load Balancing (ALB) distributes the same fraction of the workload to each server. Another approach is Best Performance to Power ratio First (BPPF). All servers are sorted by their performance to power ratio at 100 % load in a decreasing order. As long as the work is not yet distributed the most efficient server is filled up to 100 % load, then the second best and so on. Adaptive Load Distribution (ALD) is a more sophisticated approach to calculate the distribution of a specific load on n servers. The key idea is to assign each server a portion $\alpha_k \geq 0$ of the requested load. This α_k is referred to as scaling factor. The total power consumption for a load request o can be calculated by summing up the power consumption of the single servers as

$$p_{total}(o) = \sum_{k=1}^{n} p_k(o \cdot \alpha_k). \tag{1}$$

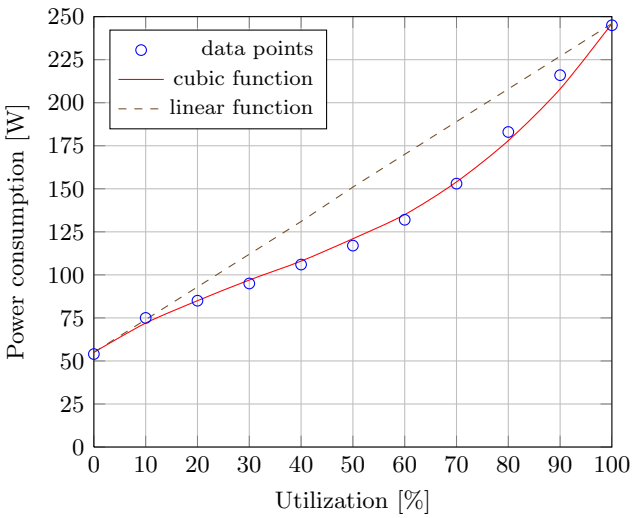

Figure 1: Linear and cubic function as model for power consumption.

Functional and implementation details can be found elsewhere [10].

ALD achieved the best overall results of all algorithms. Only in some cases BPPF was slightly better. In all cases a combination of ALD and BPPF led to the best result. The average power reduction potential is at about 10 % at best, so there is room for improvement. A modification and extension of ALD is presented in the next section.

The scaling factors α_k are constrained in the following way: $\alpha_k \cdot o \leq o_k^{max}$, and $\sum_k \alpha_k = 1$. The first constraint ensures that no server is overloaded, when o_k^{max} is the maximum possible load for that server. The second constraint ensures that the load is fully distributed. Of course, o must not exceed $\sum_k o_k^{max}$.

3.4 ALD+: An Extended Load Distribution Algorithm

To minimize power consumption a customized approach is introduced. It considers the possibility of switching off inefficient servers and recalculate the assigned load portions. This approach is based on ALD and referred to as **Adaptive Load Distribution Plus (ALD+)**.

The extension that is presented here deals with the problem of shutting down underused servers. One possibility would be to define a threshold for the scaling factors. If they are lower than the threshold and other constraints, like that the remaining systems are able to process the load demand, are met, the specific servers can be shut down after their share of work has been distributed to the remaining systems. However, as distribution schemes have to be calculated for various load levels within this task the decision for suspending or powering off nodes can be made. First, all servers with no load assigned are subjected to be powered off. Second, the servers are sorted in ascending order according to their performance to power ratio[1] at the specific load level. After that it is evaluated if the server with the currently lowest ratio can be removed (redistributing its load among the remaining servers). If the load can be processed afterwards and the corresponding energy would be lowered, the server is removed. This task has to be done in advance for fixed load levels. Several distribution schemes in regard to the different server farm usage are calculated. In that way, an energy-efficient load redistribution (e.g. changing load balancer settings) can be applied.

A simple approach to deactivate inefficient servers is based on a fixed limit, e.g. all servers with $\alpha_k \leq 0.05$ are deactivated. This involves two problems: First, in a heterogeneous environment all nodes can have different performance characteristics. Thus, the maximum value of α_k also varies and an assessment on the basis of an α_k value is not significant. Having $\alpha_n \neq \alpha_m$ may lead to the

[1]This ratio is given in operations per second per watt, which corresponds to operations per Joule

same utilization on S_n and S_m. In a worst case scenario, a server which operates at its optimum efficiency is deactivated and other, less efficient servers have to process that server's load.

Second, no constraints are considered. It has to be ensured that servers are not deactivated if the remaining servers cannot satisfy the demand. A second constraint should guarantee that a server cannot be deactivated if the total power consumption increases.

To solve this problem the results of the ALD strategy can be used as well. At a certain load level we know the optimized load distribution. The performance to power ratio for each server is calculated. After that, the servers are sorted ascending in regard to their performance to power ratio. Beginning with the server with the lowest (worst) ratio the the following algorithm is performed:

1. Set *active servers* to all servers of the server farm in ascending order in regard to performance to power ratio. Select first server.
2. Try to distribute the load on the servers excluding the currently selected server.
3. If remaining capacity is not sufficient to process load, select next server and start over with step 2.
4. If energy of configuration excluding currently selected server is higher than energy of original configuration, select next server and start over with step 2.
5. Remove server from list of active servers and adjust α values if necessary[2]. Start over at step 2.

The algorithm is repeated till the last server is processed. After that the new server configuration (set of active servers for certain phase) is available. After that step, the distribution scheme is used in conjunction with a workload trace to distribute the load. The workload trace represents the varying usage of a server farm over a certain time interval. The algorithm is depicted in Fig. 2, a thorough description is presented in Sec. 4.

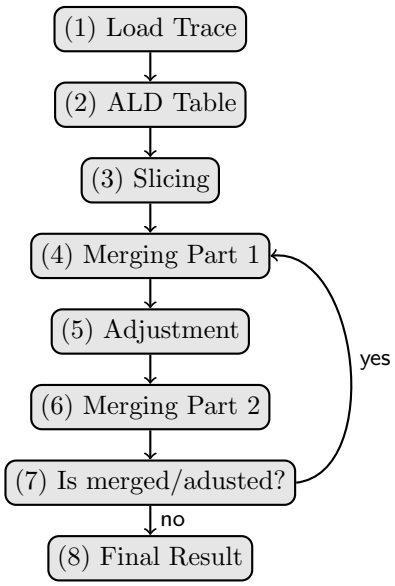

Figure 2: Algorithm for Slicing, Merging, and Adjustment

The initial node (1) represents a given load trace over a certain time frame. Within this time frame, the load may vary significantly. The ALD table according to the server configuration is calculated (2). This step is independent from the load trace and represents load distributions at different possible load levels of the server farm, e.g. in one percent steps. The first actual step of

[2]Note that this step is only executed if the conditions in steps (3) and (4) are fulfilled.

the algorithm to calculate a power off scheme is to partition the load trace into certain slices of identical lengths (3). It is beneficial to obtain big slices in which unused servers can be transferred to power saving states like suspend to main memory to save energy. To achieve that goal, in step (4) adjacent slices with the same server configuration are merged. In step (5), the slice lengths are increased by moving the edges – transitions to adjacent slices with different server configuration – if the maximum load of the slice is lower than the maximum load of the adjacent slice. The edges can be moved if the load in the adjacent slice is initially lower than the maximum load of the current slice. Step (6) considers the overhead for shutting down or suspending servers, which can be determined independently. If the energy overhead for a certain slice transition to a more power efficient configuration is higher than the benefit (energy saving), adjacent slices are merged to bigger slices in step (6). If an adjustment or merging has taken place, a reiteration of steps (4) to (7) is performed. If not, the final result containing the information about slice lengths and the distribution within the slices is available in step (8).

Regarding the performance to power saving trade off, we replayed 2000 Wikipedia requests in several rounds with different delays between the requests. Depending on the delays the overall load was considered at 10 %, 25 %, 50 %, 75 %, and 90 %. We measured the time from the start of each individual request to the arrival of the first byte of the answer as well as for the whole request include the complete answer. The mean time to the first byte is at 308 ms for RLB and 282 ms for ALD. The mean times for the request including the complete answer is 492 ms and 458 ms for RLB and ALD, respectively. The mean time to the first byte for ALD is about 8 % faster than for RLB. For the complete request, ALD is about 7 % faster than RLB.

Despite the fact that the time for an individual request may not be faster using ALD, in the average ALD performed better using the more powerful machine which additionally was more power efficient.

4 Distribution and power off considering overhead

In this section a detailed description of the application of the ALD+ approach is presented. The overhead for switching servers to different power states (idle, suspended, powered off) is considered. Above that, the task of dealing with known load curves and dividing these into suitable intervals in which the same server set is used, is discussed.

4.1 Notation

This section shows the notation used throughout this section.

i	Index for idle.
s	Index for suspended to main memory.
o	Index for powered off[3].
E	Energy in joule.
P	Power in watts.
t	Time in seconds.
$E_{i\to s}$	Energy to suspend a server to main memory.
$E_{i\to o}$	Energy to shutdown a server.
$E_{s\to i}$	Energy to resume a server from main memory.
$E_{o\to i}$	Energy to boot a server.
$t_{i\to s}$	Time to suspend a server to main memory.
$t_{i\to o}$	Time to shutdown a server.

[3]if not used as index, o is used as *overall load*

$t_{s\to i}$	Time to resume a server from main memory.
$t_{o\to i}$	Time to boot a server.
P_i	Power consumption of an idle running server.
P_s	Power consumption of a server suspended to main memory.
P_o	Power consumption of powered off server when still connected to the power grid.
$E_i(t)$	Energy consumption when running idle for t seconds.
$E_s(t)$	Energy consumption when suspended to main memory for t seconds.
$E_o(t)$	Energy consumption when powered off for t seconds.
t_{is}	Time when running idle and suspending have the same energy consumption.
t_{so}	Time when suspending and powering off have the same energy consumption.
$\varepsilon_\downarrow$	Ratio of energy for suspend to energy for shutdown.
$\varepsilon_\uparrow$	Ratio of energy for resume to energy for boot.
$\tau_\downarrow$	Ratio of time for suspend to time for shutdown.
$\tau_\uparrow$	Ratio of time for resume to time for boot.
ρ_{so}	Ratio of power in suspend mode to powered off mode.
ρ_{is}	Ratio of power in idle mode to suspend mode.

4.2 Server Information and Distribution Tables

The load distribution schemes for a certain server configuration is calculated in advance. The overall load o must not exceed the sum of the maximum loads of all n servers of the server farm, see Eq. 2.

$$0 \leq o \leq o^{max} = \sum_{k=1}^{n} o_k^{max} \tag{2}$$

Performance and power information of an example configuration used in this section is depicted in Tab. 1. This heterogeneous server farm consists of five servers that became available between June 2008 and June 2012. The system identifier (SID) in the first column corresponds to the SPECpower benchmark publication of the specific server. The second column is the maximum number of operations that the server may execute (given in SPECpower ssj_ops). The third and forth column is the maximum and minimum power consumption in watts, respectively. The last column is the month and year of the first availability of the server. A more detailed description of the servers is available from the SPECpower benchmark result information available on the SPEC website[4].

SID	o_k^{max}	P_{max}	P_i	avail
55	341600	287	156	6/2008
144	251555	294	226	7/2008
153	551440	224	54	6/2009
444	1261800	258	56	4/2012
516	1329120	286	89	6/2012

Table 1: Overview of analyzed server farm configurations.

Beside the information available or inferred from the SPEC results further parameters are necessary for the consideration of the overhead to switch inactive servers to certain power states, see Tab. 2. This parameters include energy and time overheads for the transition in or from a specific power state. As these parameters are not directly or indirectly available they were determined for

[4]https://www.spec.org/power_ssj2008/results/

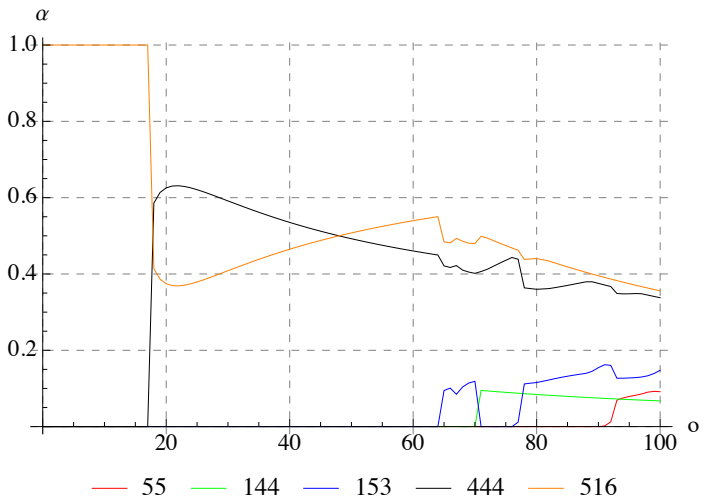

Figure 3: ALD load distribution for a five node environment.

each server. For that purpose we analyzed real servers at our laboratory. For each server, the SPECpower benchmark was executed. Above that, energy for shutting down and rebooting as well as suspending and resuming were measured. Between each shutdown and startup various usage patterns were applied, e.g. running several minutes idle between startup and shutdown or using a specific percentage of the main memory. In regard to the energy saving potential the possibility to suspend to disk was considered as well. As the results in regard to energy overhead and time were much worse than to shut down the servers, this option was omitted from the results. Despite that decision, it is possible to use suspend to and resume from disk for an actual server farm, of course. These information was used and transferred here to define the parameters for the servers used in this context.

SID	$E_{i\to o}$	$E_{o\to i}$	$E_{i\to s}$	$E_{s\to i}$	$t_{i\to o}$	$t_{o\to i}$	$t_{i\to s}$	$t_{s\to i}$	P_o	P_s	P_i
55	1804	4111	287.0	1308.7	24.2	48.2	2.0	7.6	2.1	3.9	156
144	1717	4488	338.1	1446.4	30.0	52.1	2.3	8.2	2.3	4.0	226
153	1142	3872	201.6	954.2	14.8	40.3	1.8	7.1	2.2	3.9	53
444	897	4051	141.9	975.2	9.2	28.7	1.1	6.3	2.1	4.0	56
516	874	4037	114.4	858.0	8.1	25.3	0.8	5.2	2.3	4.1	89

Table 2: Additional parameters for overhead calculations.

The following information (refer to Sec. 4.1) is given in Tab. 2: the system identifier (SID) in the first column. The next four columns represent the energy for shutdown ($E_{i\to o}$), boot ($E_{o\to i}$), suspend ($E_{i\to s}$) to, and resume ($E_{s\to i}$) from main memory in Joule. The next four columns contain the times for shutdown ($t_{i\to o}$), boot ($t_{o\to i}$), suspend ($t_{i\to s}$), and resume ($t_{s\to i}$) in seconds, respectively. The last three columns represents the power consumption when the node is powered off and still connected to the power grid (P_o), when suspended to main memory (P_s), and when running idle (P_i).

A table containing the load distribution at specific load levels is calculated applying the ALD strategy. A plot of the load distribution is shown in Fig. 3. The distribution schemes for each percent of the maximum load is used. That means that a table with 101 entries (from 0 % to 100 % load) is calculated it advance. This table includes the α values for each node at the specific overall load level. Because of the non-deterministic behavior of the ALD strategy, the distribution has to be verified and corrected after the distribution. Negative α values are omitted and the remaining values updated. Spikes, which represents unexpected and significant changes of values at a specific load level, are removed as well. In the case of the five servers introduced above, in 30 cases a negative

value occurred (mostly a very small value below $-1 \cdot 10^{-8}$, only once greater than 0.1). Two spikes at 61 % and 81 % overall load were removed.

The α values are used to calculate the operations per second assigned to a server at a specific load level, and therefore, are suitable to calculate the performance to power ratio of the server. To decide which servers are subjected to be powered off (or suspended), the servers are sorted ascending to their performance to power ratio. In this order, each server is checked: First, assuming the current server is subjected to be powered off (or suspended), it is verified if the remaining servers are able to process the load of the current server. If so, the α values of the remaining servers are adjusted to process the load demand of the server subjected to be removed from the set of active servers. The power consumption of the new set is compared to the power consumption of the set before deactivating the current server. If this energy is lower, the server is finally removed and the next server is checked. This is repeated as long as a server is removed from the list of active servers. After this procedure is performed for all load levels, a table containing the set of active servers and the load distribution (α values) is available and the next step of slicing takes place (next subsection). An extract of the load distribution, active servers, and utilization for the server farm introduced above is depicted in Tab. 3. The first column represents the percentage of the overall load o. The second column is a vector of active servers, the third column the corresponding α values. The last column represents a list of server utilization in percent.

Please keep in mind that the α value is the portion of the overall workload assigned to a specific server. So the utilization correlates only indirectly to the α value. Refer e.g to entries 33 to 35 in Tab. 3: In all entries the α value is 1.0. Despite that, the utilization changes: From 33 % to 34 %, the server is changed from 444 to 516. As server 516 is more powerful its utilization is lower at 34 % usage than the utilization of server 444 at 33 %. From 34 % to 35 % the overall load increases, the same server is used (516) and so the utilization increases on that server.

% of o	**Active servers**	**α value**	**Server utilization**
0	{ }	{ }	{ }
⋮	⋮	⋮	⋮
33	{444}	{1.0}	{97.7%}
34	{516}	{1.0}	{95.6%}
35	{516}	{1.0}	{98.4%}
36	{444,516}	{0.56,0.44}	{59.3%, 44.9%}
⋮	⋮	⋮	⋮
84	{153,444,516}	{0.18,0.40,0.42}	{99.5%,99.8%,100%}
85	{144,153,444,516}	{0.08,0.13,0.37,0.42}	{100%,76.0%,93.1%,100%}
⋮	⋮	⋮	⋮
100	{55,144,153,444,516}	{0.09,0.08,0.15,0.33,0.35}	{100%,100%,100%,100%,100%}

Table 3: Additional parameters for overhead calculations.

At 0 % load all servers can be set inactive and accordingly there is no α value. At 33 % only server 444 is used, resulting in a single α value of 1.0. The algorithm determined that a switch from server 444 to 516 is beneficial at 34 %. This is due to the fact that server 444 would be overloaded at this overall load level. An additional server would be necessary to process the load or one more powerful one, which is server 516 in that case. Above that, it is more power efficient to use only server 516 than server 444 and another server. At 36 % servers 444 and 516 are used, the α values are 0.56 and 0.44, correspondingly. At 85 % server 144 is added. At 100 % all servers are active and fully loaded.

If needed, the resolution of the distribution table can be increased (or decreased). The calculation of the α values would take more (or less) time, accordingly.

4.3 Slicing and Server State Calculation

After calculating the distribution and active server table this information can be used on arbitrary load curves. The load curve is divided into slices of the same length. We decided to use slices of 60 seconds as this is about the minimum time required to shutdown or reboot a server. These slices are merged to bigger macro-slices afterwards. An extract of an artificial load curve is depicted in Fig. 4. On the X-axis is the time in seconds (480 seconds in this example). The Y-axis represents the percentage of the overall load that has to be processed. Every 60 seconds a vertical blue line represents the transition from one slice to the next.

The blue point and red horizontal bar in each slice represent the maximum load in the specific slice. This information is used to determine which servers are used in the specific slice. As all other load points are lower or equal to that point overloading the active servers is avoided. Each slice is marked with a capital latin letter from A to H. Table 4 shows the set of active servers after the initial slicing.

Active	**Slice**							
servers	A	B	C	D	E	F	G	H
S_1	X	X	–	–	–	X	X	X
S_2	X	X	–	–	–	X	X	X
S_3	–	–	–	–	–	–	X	X
S_4	X	X	X	–	X	X	X	X
S_5	X	X	X	X	X	X	X	X

Table 4: Active servers after the initial slicing process.

The active servers are determined by the results of the ALD distribution. Servers with no load or low performance to power ratio are subjected to be removed from the list of active servers as long as the remaining servers are able to process the load demand. The distribution (α values) might be adapted during this process.

4.4 Slice Merging Step 1

The first step of the merging process is to identify adjacent slices with the same server configuration. These slices can be merged safely to bigger macro-slices. The point of the maximum is adjusted to the point of the maximum load of the merged slice. In Fig. 4 adjacent slices A and B as well as slices G and H have the same set of active servers and are therefore merged to macro-slices AB and GH, respectively. Several adjacent slices could be merged to even bigger macro-slices, of course.

In the case of the merged slice AB the new maximum load is the former maximum load of slice A. The maximum load of H would be the new maximum load of slice GH. The result is shown in Fig. 5 where six of the initial slices remain after the first merging process. Slices before A or after H are not considered in this example.

4.5 Slice Adjustment

After the first merging process the slices are adjusted to their maximum size. If a slice is not yet utilized to its maximum available load, its size can be increased, see slice D in Fig. 5. As the rightmost point inside slice C has a higher load than the maximum load of slice D it cannot be extended to the left.

While slice E has a higher maximum load, its load at the beginning is below the maximum load of slice D. Thus, slice D can be extended by moving the transition point to the right until the load is equal to the maximum load of slice D. While the size of slice D is increased, the size of slice E is decreased.

With this adjustment, the additional load of the extended slice is performed in a more energy efficient configuration set of active servers. Usually, less servers or a more energy efficient set of servers are used to perform the load demand of slices with a lower maximum load. Note, that no

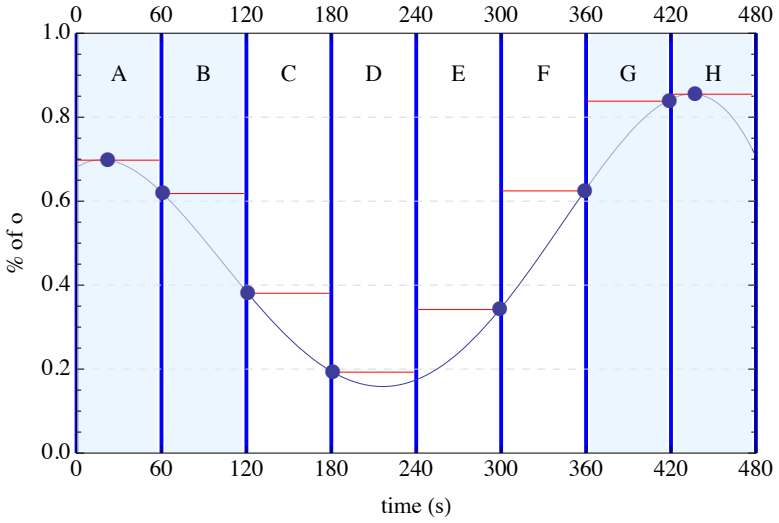

Figure 4: Slice Merging Step 1.

improvement can be achieved if the adjacent slices use the same set of servers. In this case, these slices would have been merged already in the previous step.

The result of the described adjustment step can be seen in Fig. 6 where the transition point from slice D to slice E has been moved to the right.

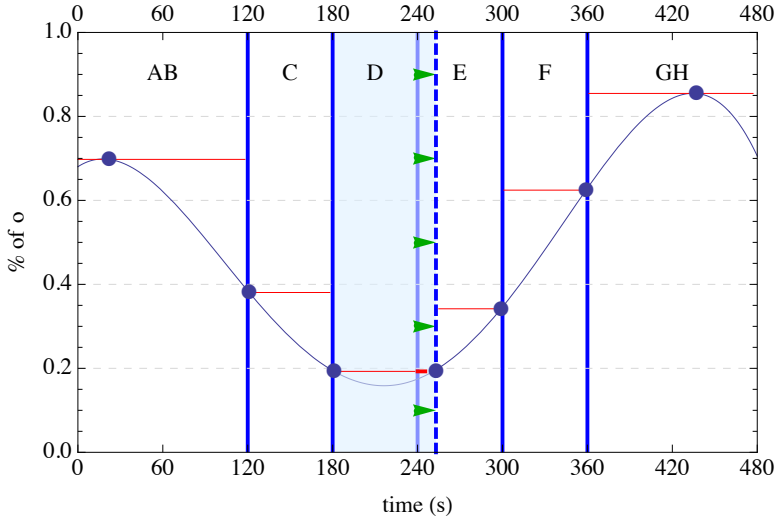

Figure 5: Adjusting Slices.

4.6 Cell Merging Step 2

Because of the overhead for shutting down or suspending servers, from an energy efficiency perspective it might be more useful to change the active set of servers to a set of the left or right adjacent slice. E.g., it might be more energy efficient to process the load demand of slice D in Fig. 6 with the set of active servers of slice C or E. In the example, the server configuration of slice C is the same as the server configuration of slice E. This may not be the case in any situation. This merging phase uses results from Sec. 4.7, which can be obtained independently from and in advance to the slicing, merging, or adjusting process. Despite the fact that results are already used, we decided to

place the description of the power state calculation after this section as it is heavily used for the calculation of the energy consumption afterwards.

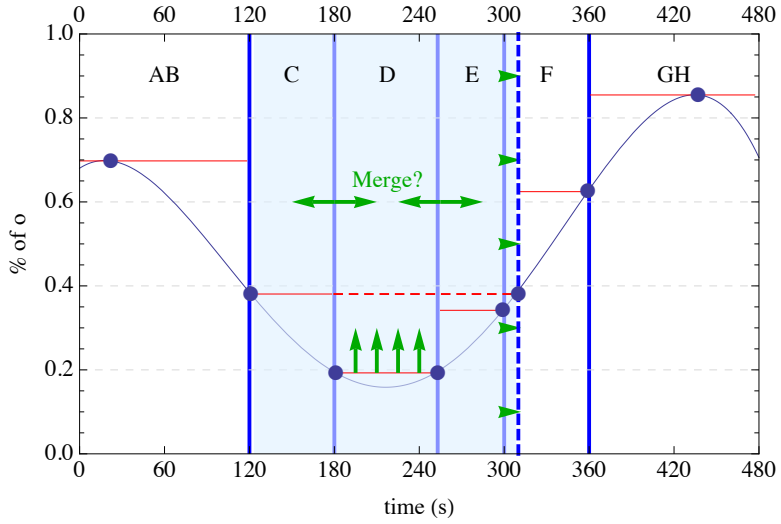

Figure 6: Slice Merging Step 2 and Adjusting.

It is assumed that the overhead for changing the server states by switching from active set C to D is higher than the benefits for processing the load demand on the original active server set of D. The energy for slice D is calculated under consideration of its active/inactive server configuration. This includes the energy consumption of the active servers processing the load demand, the energy of the inactive servers (idle, suspend, and power off energy) and the energy for the necessary power state transitions. If that energy is higher than the energy when setting the active/inactive server configuration equal to the left or right adjacent slice, the slice is merged with one of these (or both).

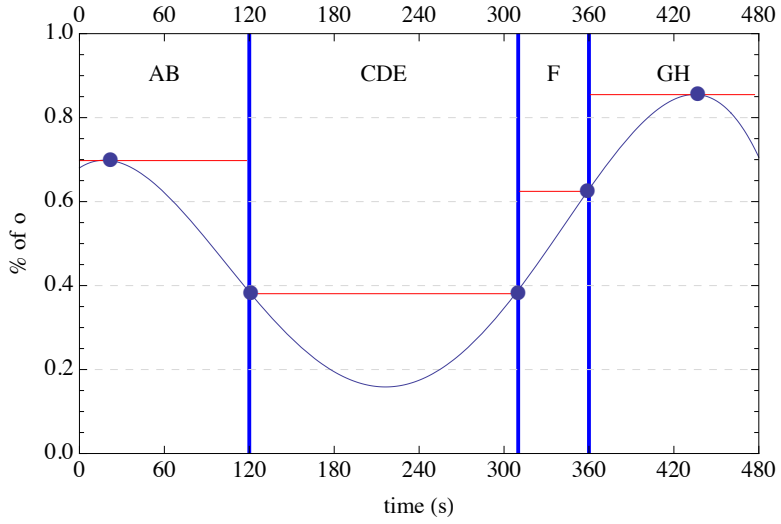

Figure 7: Final Slices.

In the considered case, slice C and D are merged to a bigger macro-slice. In the case of Fig. 6 both configurations (C and E) are equal and therefore, all three slices C to E are merged to one big cell CDE. When merging only to the left or the right slice, the server configuration of the remaining slice might be the same. In that case (as in the example) the remaining slice is merged into the new

macro-slice as well. After the merging process the maximum load might be higher than before for one of the edge slices. Therefore, a slice adjustment is carried out if needed.

The result of slice merging and adjusting is shown in Fig. 7. From the initial set of eight slices only four remain: AB, CDE, F, and GH. After merging a slice the process continues with the next slices left and right. It is examined if they can be merged with their adjacent cells. The merging process starts with the slices at a local minimum (in regard to the maximum load in the specific slice) and continues until a local maximum is reached.

Finally, a set of slices and active server states considering the overhead for transitions is constructed. This result is used to generate a state diagram for the servers with the information of activity or inactivity for each server. That graph is then used to set the state of the inactive servers: idle, suspended to main memory, or powered off. The process of merging and adjusting is repeated as long as a merging and/or adjusting takes place. In the end, the slice sizes will be maximized. In the next step the power states for the inactive servers are determined.

4.7 Power state of inactive servers

When slicing, merging, and adjusting is finished, the power states (idle, suspended to main memory, or powered off) for the inactive servers are determined. Therefore, we use the following three formulas describing the energy consumption for the three considered power states (refer also to Tab. 2 and Sec. 4.1):

$$E_i(t) = t \cdot P_i \tag{3}$$

$$E_s(t) = E_{i\to s} + E_{s\to i} + (t - (t_{i\to s} + t_{s\to i})) \cdot P_s \tag{4}$$

$$E_o(t) = E_{i\to o} + E_{o\to i} + (t - (t_{i\to o} + t_{o\to i})) \cdot P_o \tag{5}$$

Equation 3 represents the energy for letting a server run idle for a time period t. This time period can span over one or several slices. It is simply the product of t times the idle power consumption P_i of the specific server. Equation 4 is the energy consumption over a time period t when suspending this server to main memory and resuming it afterwards. It is composed of the energy for suspending and resuming ($E_{i\to s}$ and $E_{s\to i}$) and the energy consumed when suspended. The latter is the product of the t minus the time for suspending and resuming ($t_{i\to s}$ and $t_{s\to i}$) times the power consumption P_s when suspended. Equation 5 is similar to the previous and represents the case when the server is powered off (but still connected to the power grid).

By these formulas and the information about the energy and time overhead (refer to Tab. 2), the energy demands of the various power states for a specific time period in which a server may be inactive can be calculated and compared to one another. If $E_i(t) < E_s(t)$ it would be preferable to keep the server in idle mode. Furthermore, if $E_o(t) < E_s(t)$ it would be more efficient to power off the server instead of suspend it or keeping it running idle. In all other cases the server should be suspended to main memory. This comparison had to be done for all time periods a server is inactive.

But instead of recalculating the energy consumption for each phase in which a server may be in inactive state we can calculate the points in time at which two power states are equally energy efficient: The equations $E_i(t) = E_s(t)$ as well as $E_s(t) = E_o(t)$ have to be solved in respect to the parameter t which represents the time period of inactivity. The solution of the first equation is called t_{is} and is the length of an inactive phase where the energy consumption of suspending to main memory and running idle is equal. The solution of the second equation is called t_{so} and represents the length of an inactive phase where the energy consumption of powering off and suspending to disk is equal.

For time periods $t_0 < t_{is}$ is is more energy efficient to keep the server running in idle mode than to suspend it. If $t_{is} < t_0 < t_{so}$ the server should be suspended to main memory. Eventually, if $t_0 > t_{so}$ the choice would be to power off the server to save the largest amount of energy. If $t_0 = t_{is}$ or $t_0 = t_{so}$ the server can be either kept running idle or suspended in the first as well as suspended or powered of in the second case.

Before calculating t_{is} and t_{so} some dependency can be introduced between the formulas relating energy and time of shutdown/boot and suspend/resume to one another. Similarly, dependencies relating the power consumption of suspend/powered off as well as idle/suspend are introduced:

$$\varepsilon_{\downarrow} = \frac{E_{i\to s}}{E_{i\to o}} \quad \text{where } 0 < \varepsilon_{\downarrow} \leq 1 \tag{6}$$

$$\varepsilon_{\uparrow} = \frac{E_{s\to i}}{E_{o\to i}} \quad \text{where } 0 < \varepsilon_{\uparrow} \leq 1 \tag{7}$$

$$\tau_{\downarrow} = \frac{t_{i\to s}}{t_{i\to o}} \quad \text{where } 0 < \tau_{\downarrow} \leq 1 \tag{8}$$

$$\tau_{\uparrow} = \frac{E_{s\to i}}{E_{o\to i}} \quad \text{where } 0 < \tau_{\uparrow} \leq 1 \tag{9}$$

$$\rho_{so} = \frac{P_s}{P_o} \quad \text{where } \rho_{so} \geq 1 \tag{10}$$

$$\rho_{is} = \frac{P_i}{P_s} \quad \text{where } \rho_{is} \geq 1 \tag{11}$$

The energy and time for suspending and resuming are less than the energy and time for shutting down or rebooting a server, respectively. Therefore the energy and time for suspending can be expressed dependent on the parameters of shutting down a server using parameters ε and τ. Besides that, the power consumption of a suspended server is bigger than the power consumption of a powered off one. Additionally, running a server in idle mode consumes more power than when suspended or powered off (parameters ρ_{is} and ρ_{so} express this).

Before calculating t_{is}, $E_i(t)$ is substituted using the ratio ρ_{is} introduced in Eq. 11. The result of this step is shown in Eq. 12.

$$E_i(t) = t \cdot \rho_{is} \cdot P_s \tag{12}$$

After the substitution, t_{is} is calculated by solving the equation $E_i(t) = E_s(t)$ with respect to t, see Eq. 13.

$$t_{is} = \frac{E_{i\to s} + E_{s\to i} - (t_{i\to s} + t_{s\to i}) \cdot P_s}{(\rho_{is} - 1) \cdot P_s} \tag{13}$$

Similarly, $E_s(t)$ is substituted using the ratios introduced in Eqn. 6 – 10. The result is shown in Eq. 14.

$$E_s(t) = E_{i\to o} \cdot \varepsilon_{\downarrow} + E_{o\to i} \cdot \varepsilon_{\uparrow} + (t - (t_{i\to o} \cdot \tau_{\downarrow} + t_{o\to i} \cdot \tau_{\uparrow})) \cdot P_o \cdot \rho_{so} \tag{14}$$

The equation $E_s(t) = E_o(t)$ is solved with respect to t to calculate t_{so}, see Eq. 15.

$$t_{so} = \frac{(1 - \varepsilon_{\downarrow}) \cdot E_{i\to o} + (1 - \varepsilon_{\uparrow}) \cdot E_{o\to i} + ((\tau_{\downarrow} \cdot \rho_{so} - 1) \cdot t_{i\to o} + (\tau_{\uparrow} \cdot \rho_{so} - 1) \cdot t_{o\to i}) \cdot P_o}{(\rho_{so} - 1) \cdot P_o} \tag{15}$$

By the formulas of Eq. 13 and Eq. 15 it can be easily determined in which mode the server should operate when it is inactive for a time period t_0. Table 5 shows the parameters as well as t_{is} and t_{so} for the servers specified in Tab. 1 and Tab. 2.

SID	**Parameters**						**Transitions**	
	$\varepsilon_{\downarrow}$	$\varepsilon_{\uparrow}$	$\tau_{\downarrow}$	$\tau_{\uparrow}$	ρ_{so}	ρ_{is}	t_{is}	t_{so}
55	0.159	0.318	0.083	0.158	1.820	40.625	10.24 s	2427.61 s
144	0.199	0.321	0.077	0.158	1.748	56.219	7.85 s	2481.99 s
153	0.177	0.246	0.120	0.178	1.819	13.555	22.84 s	2143.58 s
444	0.158	0.241	0.122	0.225	1.905	14.000	20.91 s	1991.98 s
516	0.131	0.215	0.100	0.200	1.787	21.655	11.17 s	2124.61 s

Table 5: Parameters and transition points.

In the case of the servers introduced above it is a reasonable choice to suspend them to main memory when having an inactive phase longer than 8 to 23 seconds (depending on the server). Shutting down servers is beneficial only after a much longer idle phase of about 33 to 42 minutes. This enables decisions for energy efficient suspending with relatively short prediction of future loads (60 seconds). Powering off servers requires a more sophisticated prediction (60 minutes). In this paper we assume having a relatively good prediction of the load over a whole day. Despite that, the algorithm can cope with shorter prediction horizons if it is restricted to suspending machines to main memory. Another advantage of suspending is the fast recovery in the case of unexpected load changes. Above that, suspending to main memory keeps the state of the machine intact. For each server in a configuration the transition points defined above can be calculated once in advance and are then used to decide for an arbitrary time phase of inactivity if the server should run idle, be suspended to main memory, or powered off.

For the extract of the load trace used above the active and inactive servers over a time period of 480 seconds is shown in Fig. 8.

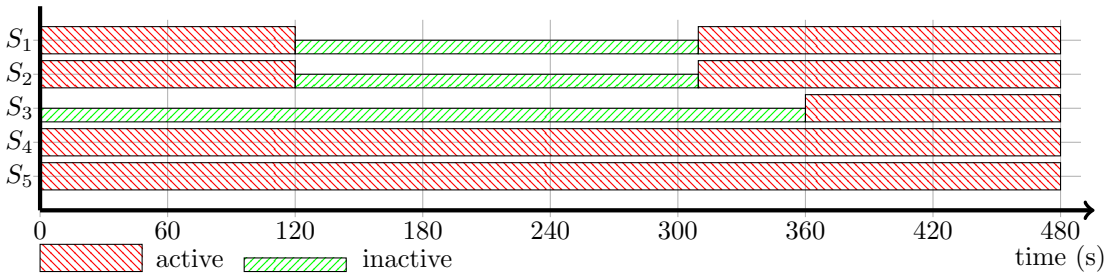

Figure 8: Active and inactive systems over 480 seconds.

Usually the time periods of the same server configuration is much longer than that of the example used above using e.g. the Wikipedia access traces. The next section deals with the final calculation of the energy consumption with respect to energy for processing the workload on the active servers, the energy consumption of inactive servers and the overhead for changing a server's state.

4.8 Energy calculation

The energy for processing the workload is calculated using the ALD distribution scheme introduced above. Because ALD does not considers shutting down or suspending servers, for each time period with a certain configuration we have to use the entry for the maximum load in that period. An alternative, with a significantly higher overhead in the preparation phase would be to calculate new ALD tables for each time period with a certain server configuration. In that case, only the distributions for the percentages of the overall load present in the particularly phase have to be calculated.

The energy consumption of the inactive servers as well as the overheads for state switching are calculated in one phase using the formulas for the energy consumption introduced above. This done independently from the configuration phases. If a server is inactive e.g. server S_3 over phases AB-CDE-F, the energy can be calculated over all these phases because no transition to another state takes place in between for that particularly server.

5 Experimental Results

Now we have a look at the results we achieved by computational simulation. In the first subsection the definition of workloads is discussed. This is especially interesting because of the fact, that for certain workloads a prediction over long time intervals is possible to some extend. This is valid for example for web access traces of Wikipedia.

In the second subsection the initial results of ALD+ without considering the overhead for state switching is described. This includes the definition of configurations. In the third subsection the results considering overhead and state switching for unused servers is presented.

5.1 Defining Workloads

We used the access traces of Wikipedia as an application of a large server farm with a varying workload during the course of the day. Those access traces contain detailed information. The hourly number of accesses and the transferred data per web request as well as the number of accesses per hour summarized over all requests are available.

We analyzed the data over a period of twelve weeks. Then, we averaged the data for each day of the week and normalized it to the maximum value. We observed differences regarding the utilization on business days and weekend days. Therefore, we decided to define two different workloads, which are displayed in Fig. 9. The green line with the triangle markers shows the daily load curve for weekend days, the dashed blue line with the circle markers for business days.

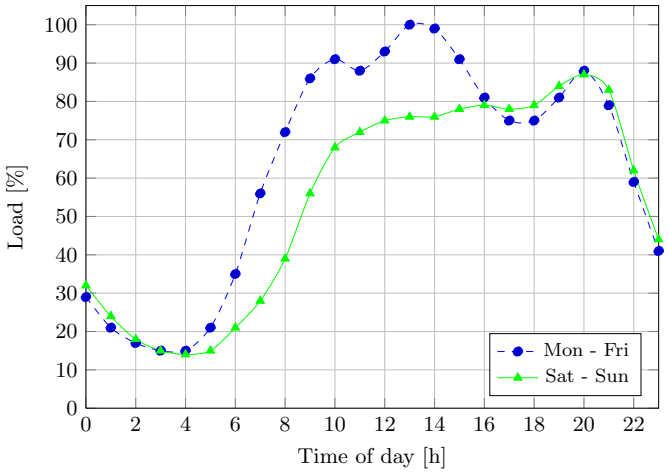

Figure 9: Workload of Wikipedia servers.

5.2 Results without considering overhead

We considered three configurations, which represent typical use cases. An overview of analyzed server farm configurations can be found in Tab. 6.

	Servers	**max. Ops**	P_i	P_{max}
Config. I	40	10,820,538	4,845	8,425
Config. II	30	21,980,959	2,204	6,183
Config. III	20	30,837,833	1,388	5,568

Table 6: Overview of analyzed server farm configurations.

Configuration I consists of 40 servers which became available about five years ago. A server farm with this configuration is able to process about $10.8 \cdot 10^6$ operations per second at a power consumption of 8,425 Watts. At active idle the power consumption is 4,845 Watts which is 57 % of the maximum power consumption. The SPEC power to performance ratio is between 450 and 1,135 ssj_ops/Watt.

Configuration II consists of 30 servers and represents the use case that the server farm has been updated by five to seven servers every year. The server hardware became available between November 2007 and December 2012. Due to the technological progress the band of the SPEC power

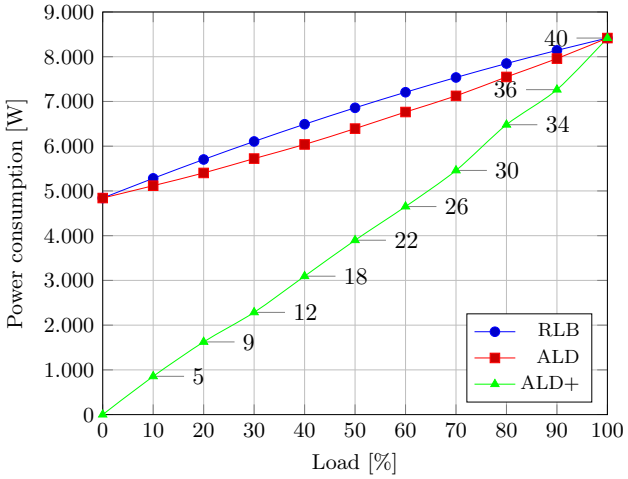

Figure 10: Power consumption of configuration I.

to performance ratio has been spread from 712 to 5,717 ssj_ops/Watt. The maximum available throughput is $21.9 \cdot 10^6$ operations per second at 6,183 Watts. The power consumption at active idle is 2,204 Watts which is 35 % of the maximum power consumption.

Configuration III consists of 20 servers, which became available from August 2012 to May 2013. The maximum throughput is $30.8 \cdot 10^6$ operations per second at 5,568 Watts. At active idle the power consumption is 1,388 Watts which is 25 % of the maximum value.

For each configuration the power consumption is calculated in three different ways. Firstly, the workload is distributed uniformly on all servers of the server farm (RLB). After that, the power consumption is calculated based on minimized α_k assuming that all servers stay active (ALD). Finally, inefficient servers may also be switched off (ALD+).

In **configuration I** there is a high potential of reducing power consumption. An optimization by ALD leads to an average reduction of 4.6 %. Especially at low utilization there is a tremendous reduction potential. At 10 % utilization 35 of the 40 servers are deactivated by ALD+. Hence, the power consumption is reduced by a factor of six. Thus, an average reduction of 41.5 % is possible. This is shown in Fig. 10. The numbers at the green graph represent the number of running servers at a specific load level.

Fig. 11 shows the load range, in which the servers run at a specific workload of the server farm. Diagram a) represents the results of ALD, diagram b) of ALD+. It is evident that many servers run in an inefficient range when using ALD only. After switching off inefficient servers, ALD+ is able to optimize the distribution towards efficient ranges. Almost all servers run either at 0 % load and are switched off, or more than 80 % load.

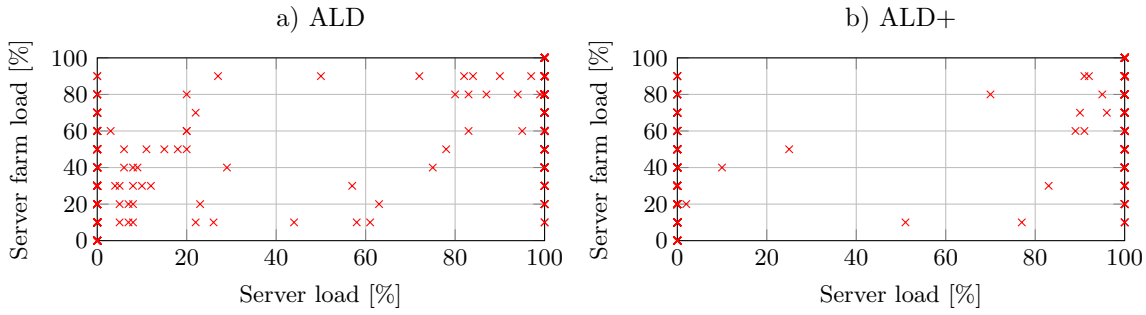

Figure 11: Scatter diagram of configuration I.

Configuration II has ten servers less than configuration I but the maximum throughput has doubled. The power consumption at active idle has been reduced by half. Fig. 12 shows a reduction potential of nearly 80 % (ALD+), particularly at low utilization. Due to the fact that newer hardware is more efficient, the third curve has an exponential shape. At low utilization, most recent servers are used. With increasing utilization more older servers are switched on. Above 90 % load the most inefficient servers are used as well.

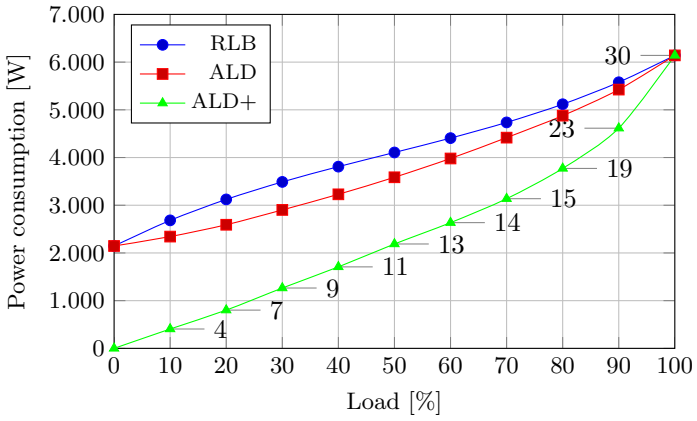

Figure 12: Power consumption of configuration II.

The scatter plot in Fig. 13 confirms a shift to a better efficiency range when servers are deactivated. As shown in diagram a), some of the servers run at approximately 20 % utilization. After deactivating inefficient servers and recalculating the load distribution, most of the servers run in an efficient range.

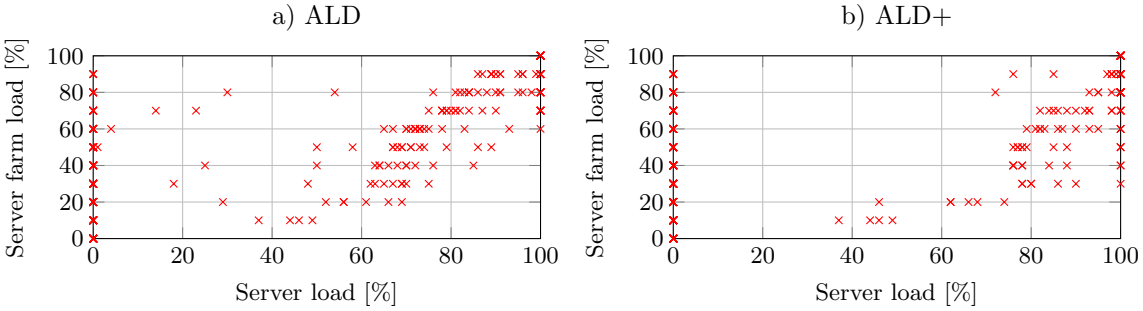

Figure 13: Scatter diagram of configuration II.

The average performance-to-power ratio of **configuration III** is very high. So power consumption is very low at high utilization. A comparison of configuration I and III clarifies the technological progress. For a throughput three times higher compared to the situation five years ago only half of the servers are needed. Fig. 14 shows the power consumption of configuration III. After minimizing α_k the efficiency was increased at all points except for active idle and full load.

The analysis confirmed the assumption that there is a significant reduction potential especially at low utilization. Therefore, inefficient servers are switched off and the load distribution was recalculated. As shown by the third curve, the power consumption converges to the curve with all servers activated as utilization increases. That is because with increasing utilization even inefficient servers have to be activated to process the desired throughput.

The scatter diagrams in Fig. 15 show the shift toward a more efficient range. Diagram a) is based on the data when using ALD. All servers run in the full spectrum from 0 % to 100 %. Diagram b) is based on the data when using ALD+. It is evident that almost all active servers run in very

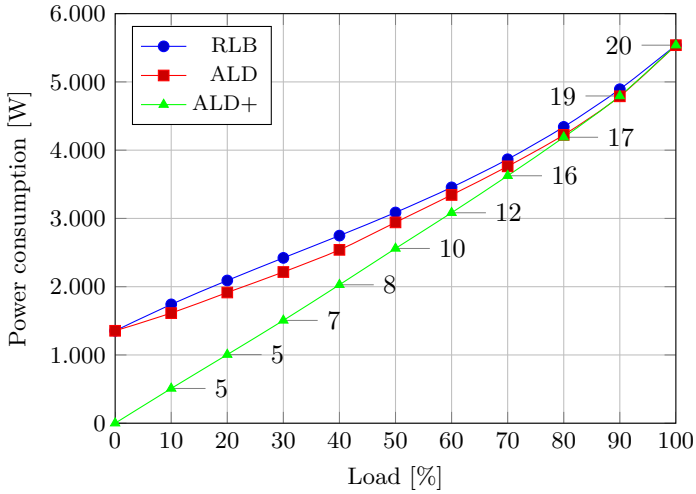

Figure 14: Power consumption of configuration III.

efficient utilization range of over 80 %.

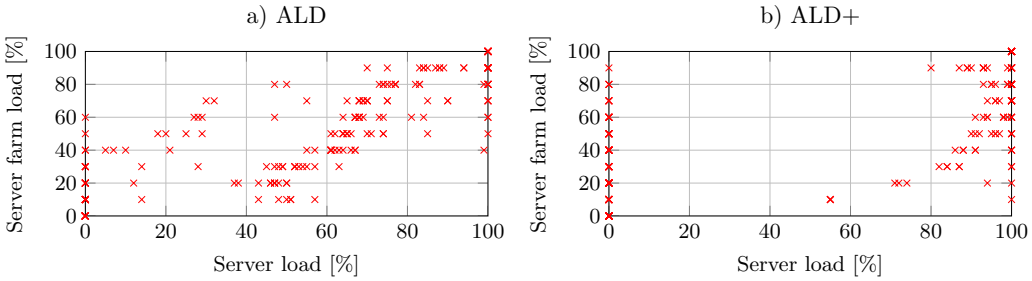

Figure 15: Scatter diagram of configuration III.

As mentioned previously, we defined two real workload curves, which are applied to configuration I. The power consumption is calculated again for RLB, ALD and ALD+. Fig. 16 show the results. RLB led to the largest power consumption for the weekday workload curves. With ALD the average power consumption was reduced by 4 %. ALD+ reduced the power consumption significantly up to 78 % especially at low utilization between 3:00 and 4:00 AM. With increasing workload ALD and ALD+ converge to RLB. The results for weekends can be found in [11].

The saving potential is illustrated not only in a context of power consumption but also in a context of economy and environment. Firstly, the Daily average Energy Consumption (DEC) is calculated. After that, a calculation for a year takes place (Annual average Energy Consumption – AEC). The economic benefit is calculated by using the Annual electricity Costs (AC). One kilowatt-hour is assumed to cost 0.1334 EUR. This kilowatt-hour produces 576 grams of carbon dioxide. Tab. 7 shows the results for RLB, ALD and ALD+. Annual electricity costs can be reduced by 20, 137 kWh, resulting in savings of 2, 687 EUR and 11.6 tons of carbon dioxide. These results does not consider further reduction due to decreased cooling demand.

	DEC	**AEC**	**AC**	$\boldsymbol{CO_2}$
RLB	171 kWh	62,572 kWh	8.347 EUR	36,04 t
ALD	164 kWh	59,999 kWh	8.003 EUR	34,56 t
ALD+	116 kWh	42,435 kWh	5.660 EUR	24,44 t

Table 7: Overview of annual electricity costs and carbon dioxide emissions.

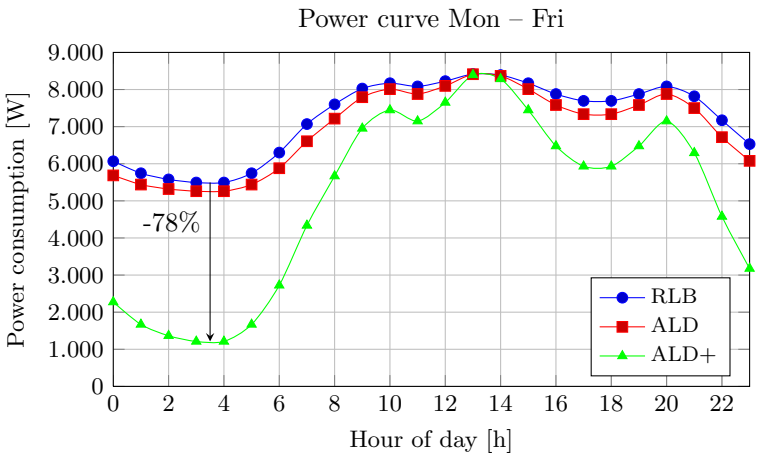

Figure 16: Workload curve Monday to Friday applied to configuration I.

5.3 Results with overhead consideration

In the previous section we addressed the basic potential of ALD in combination with shutting down hosts without considering overheads. This was conducted to decide if this approach has a significant potential to save energy. As the results were promising we refined our process including the consideration of overheads as well as the extension to suspending to memory instead of shutting down systems.

As already shown, the potential with environments consisting of up to 30 servers is quite high, we are focusing on the general results including overhead consideration with a smaller server configuration. This can be scaled easily to bigger environments. Of course, the initial calculation using ALD (possible more than once) is significantly higher when having bigger environments. As these calculations are done in advance this fact has no influence on the online behavior of our strategies.

In this section a more fine-grained load trace is used that was also derived from real Wikipedia access traces. The resolution is much higher with load information for every second of the trace. With that dense trace the energy consumption for active servers are calculated each second. The load trace is shown in Fig. 17. On the X-axis is the time of day in seconds for a whole day, on the Y-axis the load in operations per second. For the following analysis the same server configuration as in Sec. 4 is used (refer esp. to Tab. 1 and Tab. 2). The ALD results are identical as well (refer to Fig. 3 and Tab. 3).

The overall capacity of the server farm is $3.73552 \cdot 10^6$ operations per second. The load trace was adjusted to cover a range from 10 % to 80 % of the maximum possible load which corresponds to 373551 and $2.98841 \cdot 10^6$ operations per second, respectively. The server farm is slightly over-provisioned, which represents a common use case scenario.

For the slicing, merging, and adjusting process the initial slice size was set to 60 seconds. Generally, this size can be adjusted to smaller (or even bigger) values. In the mentioned case, this leads to 1440 slices for the used load curve over the time period of one day. After the initial slicing the first step of merging slices having the same server configuration took place. After that step, 353 slices remained with lengths between 60 seconds (initial slicing size) and 5400 seconds.

In Fig. 18 the number of slices for each existing slice length is depicted. On the X-axis are the slice lengths, on the Y-axis the number of slices. About half of the slices have a length of 60 or 120 seconds, the other half consists of slices with lengths up to 5400 seconds. E.g. 216 slices have been merged to 108 slices of 120 seconds length. In the case of the big 5400 second slice 90 slices have been merged.

After the merging phase, the slices were adjusted corresponding to the description in Sec. 4. This leads to slices as small as 13 seconds (one case). The biggest slice comprises a time period of 5430 seconds. Some small slices could be removed after this adjustment as the overhead for switching to

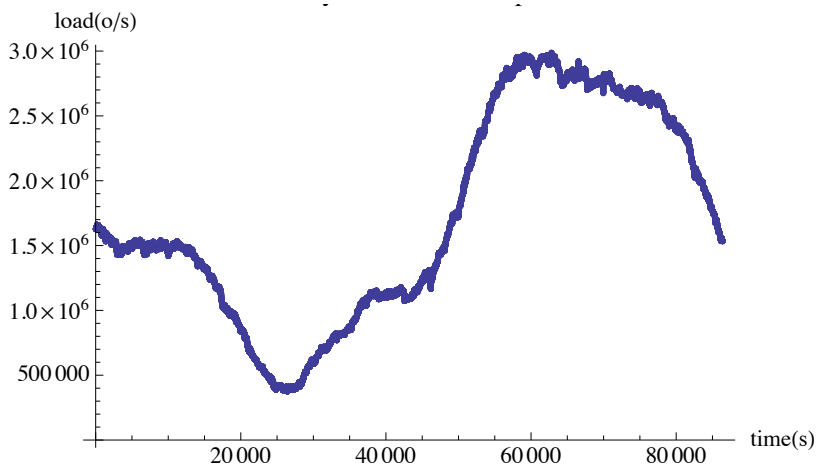

Figure 17: Load trace generated from Wikipedia access data.

another power state was bigger than the benefit gained.

The energy calculation was carried out in two phases. In phase one, the dynamic energy resulting from the active servers in each slice was calculated. Therefore, the distribution for the maximum load within the corresponding slice calculated by the ALD strategy was used. An alternative (not used in this analysis) would be to calculate new ALD tables for each slice containing distributions based on the set of active servers in the particularly slice. This would reduce the dynamic energy a little bit further.

	energy	total
active servers	34.877 MJ	–
inactive servers (idle)	36.556 MJ	71.433 MJ
inactive servers (suspended)	1.096 MJ	35.973 MJ
inactive servers (power off)	0.691 MJ	35.568 MJ

Table 8: Energy consumption for active servers and various power states of inactive servers.

In Tab. 8 the energy consumption for the active servers (dynamic) is shown in the first row. It is at about 35 MJ. In the following rows the energy consumption of the inactive servers for different power states (idle, suspended, powered off) is shown. The phases in which a server may be inactive were considered over slice borders. This means, if a server was inactive e.g. over slices 25, 26, and 27 the consideration in regard to that server covered all the three slices.

When running the servers in idle mode while no work is assigned to them the energy consumption is at about 37 MJ, resulting to a total energy of about 71 MJ including the dynamic energy of the active servers. When allowing to suspend inactive systems to main memory the energy consumption for the inactive servers is reduced to about 1 MJ. The overhead for power state switching is considered. This means, if the overhead for power state transitions to suspend is greater than the benefit for switching, the server was kept in idle state. The last power state considered is powering off inactive nodes if the overhead is not greater than the benefit for powering off. If the overhead is to big, the server was suspended or kept running idle depending on whether power state is more energy efficient. Using this power state reduced the energy consumption to about 0.7 MJ with the disadvantage of long recovery times when having unforeseen changes in the load demand. In several cases the energy overhead for switching to the powered off state was higher than the energy consumption of the suspend to main memory power state. Therefore, for that particularly time periods the corresponding servers were suspended to main memory instead of powered off.

The total energy in relation to the idle power state could be reduced to about 50 % for both suspending and powering off. The difference between suspending and powering off are quite small.

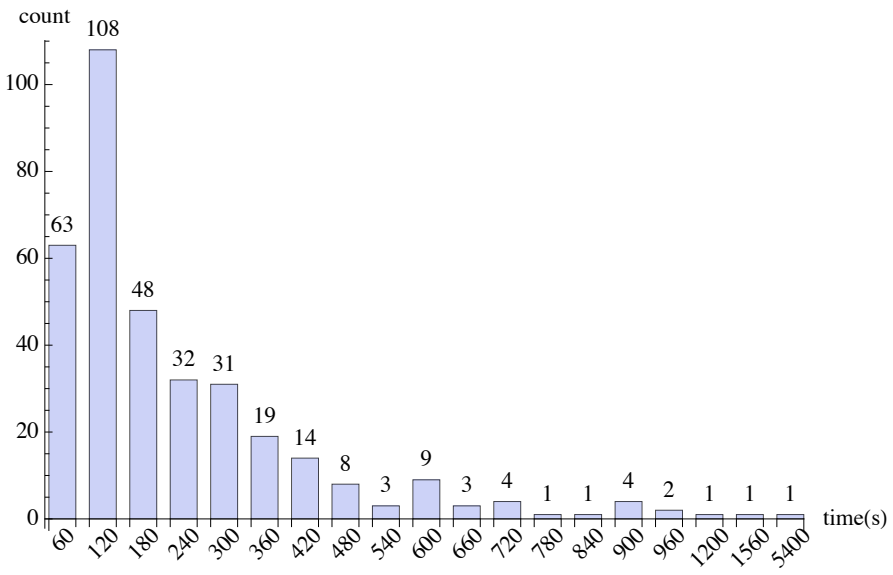

Figure 18: Slice lengths after first merging.

Because the negative aspects of powering off a server (e.g. time overhead for the transition to the active state, refer to Tab. 5) are considerably larger than suspending, it is generally a good advice to just suspend unused systems in the case of considered load trace.

6 Conclusions

This work showed that the power consumption of a heterogeneous system can be significantly reduced (up to 78 %) by an optimized load distribution and powering off scheme that reduces both heat dissipation and cooling demand. Costs and carbon dioxide emissions are lowered by 33 % because of reduced power consumption for servers and additionally by a smaller cooling demand. Furthermore, the system's reliability is increased because lower temperatures decrease the failure probability [6].

Huge power savings can be achieved by switching off servers. This is even more effective because power consumption at active idle is usually at about 50 % of the maximum value for older servers. In recent hardware this value has decreased to 25 %. Although this reduction shows the technological progress, there is still potential for power savings. Inefficient servers are completely switched off or put in a power save mode. Nevertheless, switching off inefficient servers involves overhead that was considered in this work as well. In case of an unexpected load demand, there could be a high delay till a server is active again. Especially for human-machine interaction long delays are unacceptable [20]. It is necessary to find a trade-off between low power consumption and system requirements. In modern server farms, we also have to consider the power consumption of other resources like memory and network components.

The presented strategies prove that an optimized load distribution reduces energy consumption. The algorithm to switch off inefficient servers significantly reduces the energy consumption while simultaneously considering system constraints. The scatter diagrams show a shift towards better efficiency ranges for almost all server systems. On the basis of Wikipedia traces, a realistic workload was defined and applied to all presented configurations. This reveals potential energy savings during the course of the day if the daily load curve is known. Thus, further improvements can be expected if there will be advanced load predictions schemes.

Although, this work is depending on known load curves, many ideas can be applied to on-line decisions for workload distribution and machine state changes as well when appropriately adjusted. We are currently working in this area and initial results show the potential of the approach.

References

[1] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[2] A. Beloglazov, R. Buyya, Y.C. Lee, A. Zomaya, et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2):47–111, 2011.

[3] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *NSDI 08: 5th USENIX Symp. on Networked Systems Design and Implementation*, pages 337–350, 2008.

[4] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *4th USENIX Symp. on Internet Technologies and Systems*, page 8. USENIX Association, 2003.

[5] E. Feller, C. Morin, D. Leprince, et al. State of the art of power saving in clusters and results from the EDF case study. 2010.

[6] W. Feng, X. Feng, and R. Ce. Green supercomputing comes of age. *IT professional*, 10(1):17–23, 2008.

[7] B. Guenter, N. Jain, and C. Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *IEEE INFOCOM*, pages 1332–1340, 2011.

[8] C. Hsu and S. Poole. Power signature analysis of the SPECpower_ssj2008 benchmark. In *Int. Symp. on Performance Analysis of Systems and Software*, pages 227–236. IEEE Computer Society, 2011.

[9] Y. C. Lee and A. Y. Zomaya. Energy efficient utilization of resources in cloud computing systems. *Journal of Supercomputing*, 60(2):268–280, 2012.

[10] J. Lenhardt, W. Schiffmann, P. Eitschberger, and J. Keller. Power-efficient load distribution in heterogeneous computing environments. In *12th IASTED International Conference on Parallel and Distributed Computing and Networks*, 2014.

[11] J. Lenhardt, P. Stellmach, and W. Schiffmann. Load Distribution and System Shutdown for Power-Efficient Computing in Heterogeneous Server Farms. In *3nd Int. Symp. on Computing and Networking*. IEEE, 2014.

[12] C. Liu. Energy-efficient workload mapping in heterogeneous systems with multiple types of resources. In *Proc. of the 21st Int. Conf. on Parallel architectures and compilation techniques*, pages 491–492. ACM, 2012.

[13] M. Poess, R. O. Nambiar, and K. Vaid. Energy benchmarks: A detailed analysis. In *1st Int. Conf. on Energy-Efficient Computing and Networking*, pages 131–140. ACM, 2010.

[14] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. *Power Aware Computing and Systems*, 2008.

[15] R. Schöne, D. Molka, and M. Werner. Wake-up latencies for processor idle states on current x86 processors. *Computer Science - Research and Development*, pages 1–9, 2014.

[16] J. M. Sola-Morena, K. Gilly, and C. Juiz. An approximation of energy efficiency in web systems. *Procedia Computer Science*, 18(0):2595 – 2598, 2013. Int. Conf. on Computational Science.

[17] SPEC. *SPEC — Power and Performance, Design Document, SSJ Workload, SPECpower_ssj2008, rev1137*, 2012.

[18] G.L. Valentini, W. Lassonde, S.U. Khan, N. Min-Allah, S. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, et al. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, pages 1–13, 2013.

[19] G. Varsamopoulos and S. Gupta. Energy proportionality and the future: Metrics and directions. In *39th Int. Conf. on Parallel Processing Workshops*, pages 461–467. IEEE, 2010.

[20] Q. Zhang, M.F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J.L. Hellerstein. Dynamic energy-aware capacity provisioning for cloud computing environments. In *9th Int. Conf. on Autonomic Computing*, pages 145–154. ACM, 2012.