Proceedings of the IASTED International Conference
Parallel and Distributed Computing and Networks (PDCN 2014)
February 17 - 19, 2014 Innsbruck, Austria

POWER-EFFICIENT LOAD DISTRIBUTION IN HETEROGENEOUS COMPUTING ENVIRONMENTS

Joerg Lenhardt and Wolfram Schiffmann
Computer Architecture Group
FernUniversität in Hagen, Germany
Joerg.Lenhardt@FernUni-Hagen.de
Wolfram.Schiffmann@FernUni-Hagen.de

Patrick Eitschberger and Joerg Keller
Parallelism and VLSI Group
FernUniversität in Hagen, Germany
Patrick.Eitschberger@FernUni-Hagen.de
Joerg.Keller@FernUni-Hagen.de

ABSTRACT
High performance servers of heterogeneous computing environments, as can be found in data centers for cloud computing, consume immense amounts of energy even though they are usually underutilized. In times when not all computing capabilities are needed the task to be solved is how to distribute the computational load in a power-efficient manner. The question to be answered is, what load partitions should be assigned to each physical server so that all work is done with minimal energy consumption. This problem is closely related to the selection of physical servers that can be switched off completely to further reduce the power consumption. In this work, we present algorithms which calculate a power-efficient distribution of a divisible workload among multiple, heterogeneous physical servers. We assume a fully divisible load to calculate an optimized utilization of each server. Based on this distribution, an iterative process is carried out to identify servers, which can be switched off in order to further reduce the power consumption. With that information, workload (re)distribution can take place to partition appropriate subloads to the remaining servers. As before, the calculated partitioning minimizes the power consumption.

KEY WORDS
Power-efficient compute load distribution; switching off servers; daily load curve; minimizing energy consumption

1 Introduction

In modern data centers one typically finds hundreds or even thousands of high performance servers. In many cases these servers are highly virtualized, meaning that on few physical servers a hundred or more virtual machines reside. For example, this situation is often found in cloud computing environments where hard- and software components are unified in a pool of resources that can be rented on demand. In these environments a machine consisting of operating system and application software is no longer bound to a specific computer hardware but can easily be moved between physical machines. Especially in this case a balancing of load is easily achievable. Depending on the provision model either software or hardware or both of them will be provided as Software, Platform or Infrastructure as a Service (SaaS, PaaS or IaaS). In order to maximize the providers' profit and simultaneously reduce carbon dioxide emissions, HPC objectives changed from pure performance to power-efficient solutions which maximize the ratio of a requested rate of operations to the electrical power needed to provide it.

The virtualization approach directly addresses the issue of underutilized server hardware which was very common some years ago. Today it is possible to utilize computing hardware more efficient, which was used sometimes less than 10% in former years. Computing demands may vary over time. For example, in the morning hours a peak performance demand of up to 90 or 100% is quite common, over the day less performance is needed while in the night hours only 10 to 20% of the total capacity is used. Other configurations might have other time intervals and different performance requirements. Fluctuations of the demands depend on the considered period of time. While they can vary strongly over long periods their fluctuation becomes smaller during short time intervals.

In our work we examine the idea to precalculate power-optimized load-balancing schemes for server farms to reduce power consumption at different levels of performance demand. A power optimization approach for heterogeneous cluster computing systems will be presented that distributes a divisible computational workload to a heterogeneous collection of servers. For several load request levels four different distribution schemes are considered. The best solution gives us an indication how to distribute the actual (non-divisible) load. Coarse-grained loads, i.e. few non-divisible tasks having a heavy load, have to be distributed with techniques like bin packing to achieve the precalculated load distribution as exactly as possible. Fine-grained loads, that means many tasks with a small load required by each, can almost be treated as a divisible load.

Virtualization techniques offer the possibility to easily move software components between hosts in just a few seconds. The overhead of movement regarding downtime (often almost equal to zero) and additional power consumption is negligible as we intent to hold a specific partitioning over several hours. We can save much more energy over this time period than is used for the migration of virtual machines. The VM technology relaxes the assumption of a divisible load. Being not fully divisible a server with one

DOI: 10.2316/P.2014.811-015

hundred virtual machines, each only using about one percent resources (CPU) gives us the opportunity to handle these virtual machines almost as a divisible load. Having fewer virtual machines with higher demands is also manageable, but in this case techniques like the above mentioned bin packing for distributing the load in a power-efficient manner might be necessary. One of our objectives is to guarantee that the performance needs are always ensured while applying the power saving methods. All calculations regarding the optimized load distribution take place in advance for specific load levels (percentage of the overall maximum performance capability of the server farm).

An optional power-off algorithm to identify servers that can be switched off at a specific load level is presented for one of our strategies. Although not applicable in some cases, the switching off of computing hardware offers a huge power saving potential. This is because of the power consumption of idle servers. For the fixed set of servers in a data center and various load levels, we can precalculate and store the optimized scaling and/or switch off patterns in a lookup table. Thus, the control of load partitioning can be accomplished in real time.

The power-aware scaling approach was able to partition the workload to a collection of heterogeneous servers while retaining the requested performance. It turned out that by switching off servers a power reduction of 80% can be achieved in comparison to the initial partitioning.

The remainder of this paper is organized as follows: Related work is discussed in section 2. In section 3, we present our power minimization approach as well as the switch off algorithm. Experimental results are presented in section 4. We conclude and present ideas for future work in section 5.

2 Preliminaries

In this section we give a brief overview of the power benchmarks in general and especially the used SPECpower_ssj2008 benchmark. Then the related work is presented.

2.1 Power and Energy Benchmarks

Poess et. al. analyzed several energy benchmarks [12], among them Transaction Processing Council (TPC) benchmarks, Storage Performance Council (SPC) benchmarks and the SPECpower_ssj2008.

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 consumption 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. SPC/1C targets on the comparison of smaller configurations and component-level storage products. 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 benchmarks SPECpower_ssj2008 was initially developed for power/performance analysis by the Standard Performance Evaluation Corporation (SPEC). It targets a single server or a combination of identical servers. Since 2008, SPEC published over 400 benchmark results of different server configurations. As our main interest is the power/performance ratio of single server systems 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 SPEC power benchmark[1], retrieved Jan 9, 2013. SPECpower relies on Server Side Java (SSJ) for measuring power consumption of servers at different load levels running Java applications [14]. Current servers include technologies which reduce the power consumption if the system is not fully utilized. Many of these technologies are recognized by the SPEC power benchmark.

Performance and power consumption are measured at different discrete levels of system load. 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 [14]. There are two phases in an SSJ run: A calibration phase, and 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% is referred to as *active idle*, which means that the application can accept requests, but none are actually received.

2.2 Related Work

Power consumption of recent microchips must be kept within reasonable bounds [2]. Thus, chip architects provide means to scale the supply voltage. If voltage is reduced, frequency has to be reduced as well due to increasing signal delays by the underlying logic. This Dynamic Voltage and Frequency Scaling (DVFS) is used in most contributions towards power-efficient use of modern computer hardware [5]. Power management can be implemented on

[1] http://www.spec.org/power_ssj2008/results

node level [8] or on cluster level [6] that belongs also to our approach. Moreover, low-power and power-aware computing can be distinguished. While the former one relies on a large number of low-power processors [15], we are focusing on the second approach where the power consumption is controlled usually by means of DVFS [4, 17].

In contrast to the cited DVFS-based algorithms our approach relies on the power saving capabilities which are provided by the servers' firmware and in combination with the operating system. Thus, besides the CPU also other computer components like memory and hard disks are controlled by the node's built-in power management system. Our work extends this server provided management system by workload scaling and switching capabilities for power-efficiency. Similar to [10] it assumes divisible loads that are assigned to the available servers. The specific load partitions could consist of virtual machines that are moved by live-migration between the servers [16].

Pinheiro et. al. proposed a load balancing and unbalancing algorithm for power and performance in cluster-based systems referred to as Load Concentration (LC) [11]. The primary objective is power conservation, the secondary goal of saving energy is achieved as well. Regarding the high idle power of server systems and the small difference when utilized at full load, Pinheiro et. al. did not consider load balancing schemes between nodes. Systems are switched off to save energy while their load is assigned to the remaining hosts. This is our optional step after distributing load between servers to optimize power consumption without switching off nodes. While [11] only considers homogeneous cluster nodes, our approach can also manage heterogeneous clusters.

Weisberg and Wiseman explored an approach to reduce power consumption of memory in avionic and embedded systems by switching off unused hardware components [18]. First the memory portions that are actually used are identified, then the remaining portion can be switched off temporarily. When needed they are turned on again.

Bianchini and Rajamony examine different power management mechanisms like Dynamic Voltage Scaling (DVS) or halting/deactivation [1], which provide substantial savings in power. Energy management techniques for storage servers and clusters are presented. The authors provided insight in at that time current and future work: The energy conservation strategies for application servers are explicitly mentioned as then not considered yet.

Our contribution consists mainly in modeling the total power consumption based on recent SPECpower_ssj2008 measurements and the partitioning of a given workload to a collection of commercially available servers. Moreover, we devised a switching off algorithm that can further reduce the power consumption already achieved by the partitioning. The switching off approach identifies redundant servers and redistributes their load fractions to the remaining nodes. By switching off those servers, their idle power can be saved which results in a significant reduction of the overall power consumption.

3 Optimized Power Configuration

In this section we present the key elements of our algorithms. Starting with an overview, the second subsection deals with cubic fitting of data representing electric power consumption for a specific load of a server system. The third subsection describes the central idea of minimizing the overall power consumption of a server farm at a given performance load. In the fourth subsection an algorithm is proposed to further reduce the power consumption by switching off servers and redistribute their load to the remaining systems. The fifth subsection deals with the distribution of load with a certain granularity.

3.1 Overview

The basic idea is to minimize the power consumption by partitioning the load over a server farm. To achieve that, we need information relating different load levels of a server to the power consumption on that levels. This information is already available for hundreds of popular servers and for not yet benchmarked systems it could be retrieved by the SPECpower_ssj2008 benchmark. For each server we have eleven data points at different load levels given as ssj_ops (server side Java operations per second [7]) and their corresponding power consumption in watts. The data is spread across active idle load (the system is running, the benchmark application is ready to process data but no computations take place) to 100% load in 10% steps.

As these data points are discrete values, a fitting algorithm calculates a cubic approximation of a power function p_i for each server S_i. This power function is then used by different strategies to calculate load partitions over a set of servers.

3.2 Power Curve Fitting

We relied on cubic curve fitting as we observed during experiments that the deviation is relatively low compared with the data retrieved by SPEC power benchmark. The power function for a single server S with load x is

$$p_s(x) = ax^3 + bx^2 + cx + d \tag{1}$$

The data itself used for fitting can be every conceivable combination of operations per time unit with respect to power consumption, e.g. MIPS and watts or, as in our work, based on ssj_ops and watts. As parameter the function $p_s(x)$ takes the operations per time unit o_s that server S should perform. The result is the power consumption of the specific machine. The restricted domain is limited to reasonable inputs. These have to be greater than 0 as negative operations per time unit do not make sense. The input must not exceed the maximum of operations that the system is able to perform per time unit. So x has to be in a given range

$$x \in \mathbb{R}, [0 \leq x \leq o_S^{max}], \tag{2}$$

where o_S^{max} is the maximum number of operations the system can perform per second, i.e. when running at 100% load.

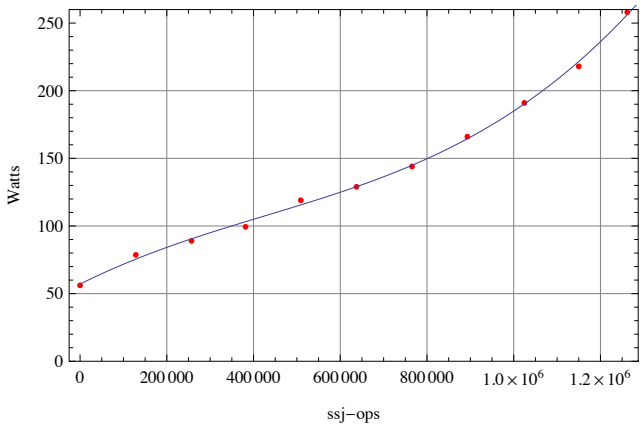

Figure 1. Curve fitting of example server S_1.

The results of the fitting process for an example system S_1 is given in Fig. 1. Server S_1 consists of two Intel Xeon E5-2660 processors running at 2.2 GHz. Each chip has eight cores, resulting in 16 cores overall. Each core has both, 32 KB of L1 instruction cache and another 32 KB of L1 data cache, 256 KB of unified L2 cache per core and 20 MB unified L3 cache. 24 GB of RAM are available in total. Tab. 1 contains the benchmark data for S_1, consisting of load, corresponding ssj_ops and power consumption. Besides that, the power consumption calculated with our power function and the error in relation to the SPEC result are given, both rounded.

		Ø-power		
Load	**ssj_ops**	**SPEC**	**func**	**error**
98.8%	1,261,803	258 W	256 W	-0.71%
90.0%	1,149,911	218 W	221 W	1.63%
80.2%	1,024,508	191 W	190 W	-0.40%
69.9%	893,029	166 W	164 W	-0.93%
59.9%	765,556	144 W	145 W	0.60%
49.9%	637,441	129 W	129 W	0.02%
39.9%	509,042	119 W	116 W	-2.87%
29.9%	381,572	99.4 W	103 W	3.79%
20.1%	256,980	89.0 W	90.5 W	1.70%
10.1%	128,555	78.6 W	75.6 W	-3.93%
Active idle	0	56.1 W	57.0 W	1.62%

Table 1. Load, ssj_ops and average power consumption of example server S_1.

The absolute maximum error of the cubic fitting function for the example server S_1 is a bit lower than 4% while the average error is about 1.6%.

Rivoire et. al. denoted that a model is accurate enough if the average error is lower than 10% [13]. With our cubic model the average error for all benchmark data is less than 4.3%, in over 80% of the cases less than 3%. Above that, the maximum error is less than 10% in 85% of the cases. The overall maximum error is about 13.8%.

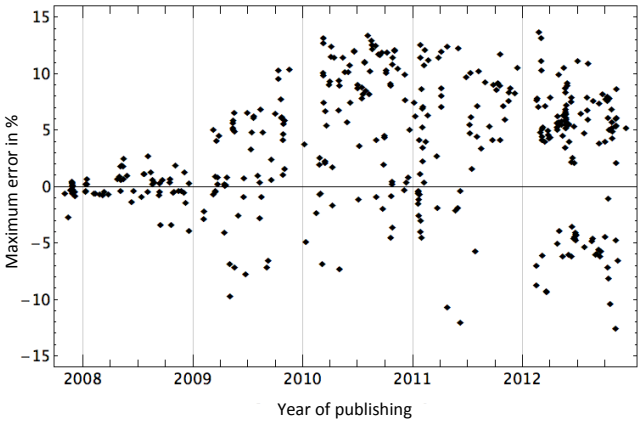

Figure 2. Maximum error of fitting process

Fig. 2 shows the trend of the maximum error for all benchmark results from late 2007 to late 2012. Each data point represents the maximum error of the cubic fit on a specific benchmark result. Until early 2009 the maximum error was at about 5% absolute. From then to early 2010 the error increased to 15% and stays at this level afterwards. Hsu and Poole observed similar results for the linear fit which provided acceptable results before 2010 [9]. The error was at about 10% increasing to nearly 40% since 2010.

Hsu and Poole presented a power signature analysis of the SPECpower_ssj2008 benchmark relying on different fitting approaches for SPEC data up to 2010 [9]. With newer servers and better power management features we observed an increasing maximal and average error. As modern servers usually depend on DVFS for reducing power we chose the cubic fitting model, because a cubic model is necessary to model reality. Power consumption increases linearly by voltage and quadratic by clock rate. As supply voltage and clock rate correlate, a cubic relation must be assumed when using power and energy saving mechanisms like DVFS [3].

3.3 Strategies for Optimizing Power Consumption

We developed four strategies to distribute a given load in a power-efficient manner on a collection of heterogeneous servers. We concentrate on a collection of servers which process a given computational demand of o operations per second.

Relative load balancing (rlb) distributes the load o to all systems in such a way that all servers are equally loaded. If o is 10% of the maximum load, each system is assigned a fraction of o so that it runs at 10% load.

Absolute load balancing (alb) distributes o so that all systems process the same amount of operations per second. This works fine till the least powerful system is fully

loaded. Then the remaining load is equally distributed among the remaining systems and so on.

Best performance to power ratio first (bppf) sorts the systems in decreasing order according to their performance to power ratio at 100% load. At first the system with the highest performance to power ratio is filled up to 100% load. After that, the system with the second highest ratio is used and so on until o is zero.

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 $\alpha_i \geq 0$ of the requested load. The total power consumption of the load request o can be calculated by summing up the partial power requests of the different servers as

$$p_{total}(o) = \sum_{i=1}^{n} p_i(o \cdot \alpha_i). \quad (3)$$

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

The next step is to calculate a set of α_i values so that Eq. 3 is minimized. As a result the load distribution consisting of α_i values is retrieved. It determines the distribution of the requested load to subloads $\alpha_i o$ on the servers S_i.

In Fig. 3 the power functions of two servers are depicted with their original SPECpower data as crosses. Obviously, server S_1 (black dashed line) has a lower power consumption than server S_2 (gray line). As the calculations for *ald* are not influenced by the idle power, both power curves are shown in normalized form (thin lines). Note, that the power consumption of S_2 increases faster than that of S_1. The maximum load of S_1 is reached at $530,326$ ssj_ops (180 W), of S_2 at $900,762$ ssj_ops (386 W).

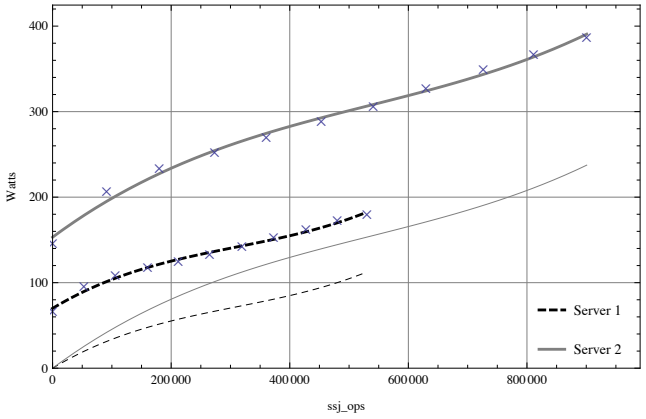

Figure 3. Fitted power functions of servers S_1 and S_2.

Fig. 4 shows the development of the α_i values and subloads for both servers. The black and dark gray dots represent the α_i value and subload for server S_1, the crosses and squares for server S_2. We can identify three phases: Phase 1 is between 0 and about 37.5% of the overall load. In this phase the α value for S_1 is 1 all the time (this leads to an alpha-value of 0 for S_2). The load of S_1 increases linearly until the system is fully loaded at about 37.5% maximum load while S_2 is idle. If we look at the power curves for both systems the power consumption of S_1 is rising less than that of S_2.

Phase two is between about 37.5 and 54%. In this phase S_1 is already fully loaded. Increasing the overall load further above 37.5% leads to utilize S_2. The load of S_1 keeps stable at 100% and the load of S_2 increases linearly to about 25% at the end of phase two. The α value of S_1 decreases while the α value of S_2 increases. Thus, S_2 must provide compute power for the increased load.

Phase three is between about 54 and 100%. This phase is most interesting because now the algorithm does not continue to put load on S_2 and keep S_1 fully loaded. It seems beneficial to decrease the load of S_1 and assign even more work on S_2. If we take a look at the normalized power curves this behavior can be explained: At first the derivative of S_1 is less than S_2. So all work is assigned to S_1 (phase one). After that, S_2 has to be used. The derivative of S_2 is higher in the beginning, but gets smaller over time. At a certain load the derivative of S_2 gets lower than that of S_1 at 100% load. If this happens, it is more power-efficient to remove load from S_1 and assign more load to S_2. This is exactly what happens in phase three. The α value of S_1 decreases faster while the α value of S_2 increases faster. At some point of time (at about 65%) the system load of S_1 increases again. Finally both systems are fully loaded at 100% overall load.

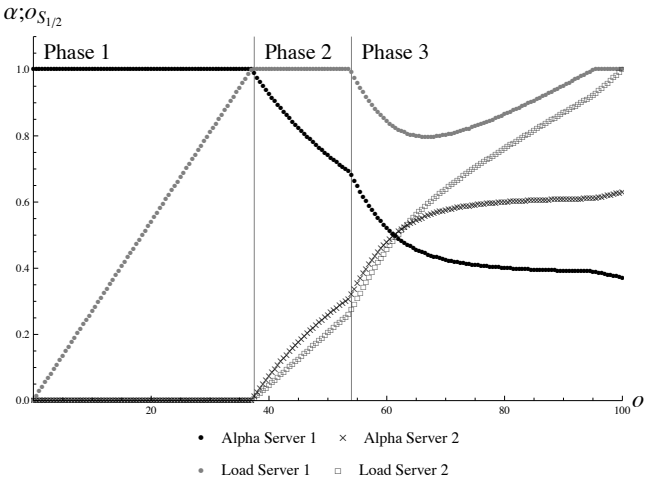

Figure 4. Development of α-values for servers S_1 and S_2.

This example shows that it is quite useful to distribute the work among the two servers according to the calculated values of α. There are situations where a combination of servers and a specific input o lead to α_i values that are 0 or close to 0. In these cases, if possible, a shut down of the corresponding servers might be beneficial.

3.4 Switch-off Algorithm for Adaptive Load Distribution

After computing all α_i values there are two possibilities to reduce the power consumption: (1) Switching off servers with α_i close to 0 or (2) switching off servers and distribute their load to the remaining servers as long as a lower power consumption is achieved. The first possibility occurs if a server is notably less energy-efficient than the others, which results from the convexity of the power function. Thus, even in heterogeneous systems servers with high energy efficiency are served first, and others are considered only if the load is high enough to completely fill the others. The second possibility is not covered by the minimization process as it does not consider to switch off servers.

An algorithm that finds an optimal solution has to consider all possible combinations of (on, off) for all n servers, which leads to a complexity of $O(2^n \cdot m)$, where n is the number of servers and m is the complexity for the minimization process.

In Fig. 5 an algorithm for further power reduction is depicted which has a complexity of $O(n^3 \cdot m)$, where n is the number of servers, m is the complexity of the minimization process and Θ is a threshold of the minimal utilization. When falling below Θ a system is switched off and the residual load is distributed among the remaining systems.

In line 1 a set of servers S_{init} is defined. This set contains all servers to consider and all information needed to perform the minimization process, i.e. all power functions p_i. It is assumed that the operations per second to be performed o is known all over the algorithm. In line 2 an initial minimization run is performed by calling the procedure `CalculateAlphas`. This procedure implements the approach illustrated in subsection 3.3. The results are assigned to the set A_{init}. After that the procedure `ReducePower` is called in line 3 which performs the additional power reduction.

The procedure `ReducePower` in lines 4 to 31 takes two parameters: S is the initial set of active servers and A the initial set of α values. It returns a reduced set of servers which perform the assigned work more power-efficiently and the α values representing the work distribution.

The outer loop from lines 5 to 30 is repeated as long as an improvement in power reduction can be achieved. The loop in lines 6 to 12 removes all servers from S whose α values are smaller than a threshold of Θ. All servers which have no or only a small amount of work assigned are switched off. In our experiments we set $\Theta = 0.01$ or 1% of o. In line 13 the power consumption of the current set of servers is calculated. The procedure `PowerDemand` implements Eq. 3. The variable `isPowerReduced` is set to `false` in line 14.

The next loop from lines 15 to 25 successively removes servers from S. The temporal set is assigned to S_{tmp} in line 16. After that, the α values for the reduced set are computed and assigned to the variable A_{tmp} in line 17. The

```
1:  S_init ← set of servers S_1 to S_n
2:  A_init ← CALCULATEALPHAS(S_init)
3:  (S_final, A_final) ← REDUCEPOWER(S_init, A_init)

4:  procedure REDUCEPOWER(S, A)
5:      repeat
6:          for i ← 1, S.length do
7:              if A[i] < Θ then
8:                  REMOVE(S, i)
9:                  REMOVE(A, i)
10:                 A ← CALCULATEALPHAS(S_tmp)
11:             end if
12:         end for
13:         p_min ← POWERDEMAND(S, A)
14:         isPowerReduced ← false
15:         for i ← 1, S.length do
16:             S_tmp ← S \ S_i
17:             A_tmp ← CALCULATEALPHAS(S_tmp)
18:             p_tmp ← POWERDEMAND(S_tmp, A_tmp)
19:             if p_tmp < p_min then
20:                 p_min ← p_tmp
21:                 S_new ← S_tmp
22:                 A_new ← A_tmp
23:                 isPowerReduced ← true
24:             end if
25:         end for
26:         if isPowerReduced = true then
27:             S ← S_new
28:             A ← A_new
29:         end if
30:     until isPowerReduced = false
31: end procedure
```

Figure 5. Iterative *Switch-Off* algorithm for further power reduction.

power consumption for the reduced set is calculated and assigned to p_{tmp}. If the temporal power consumption p_{tmp} is less than p_{min}, p_{min} is set to p_{tmp} in line 20. Above that, the temporal sets S_{tmp} and A_{tmp} are assigned to S_{new} and A_{new} in lines 21 and 22. In addition, `isPowerReduced` is set to `true` in line 23.

The loop is iterated for all servers in S. After that, if a better solution was found, S_{new} and A_{new} are containing the servers and α values for the best solution. In that case these new sets are assigned to S and A respectively in lines 26 to 29 and the outer loop is reiterated. It has to be ensured that at least one server remains in the set of servers S. It also has to be ensured, that no server is overloaded and all work is distributed on the remaining set.

3.5 Load Distribution at a Certain Granularity

If load is distributed with a certain granularity g, then for each system one can alternatively generate an array of length o_i^{max}/g where $a_i[j] = p_i((j+1)g) - p_i(jg)$ de-

notes the additional power to increase the load from jg to $(j+1)g$. These arrays are then merged like in merge sort (where each entry also contains i and j), so that the first k entries contain information on how to distribute a load of kg onto servers in the most power-efficient way. A nice side effect is that servers not referenced by the first k elements can be switched off. The granularity used can be adapted to application needs. Also, if g is known in advance, precomputations for several load levels, i.e. several values of k, could be done so that the load fractions α_i can be precomputed as well. The complexity of the algorithm is $O(o/g)$ without precomputation, so it can be accelerated by using coarser grain, and $O(1)$ with precomputation.

4 Discussions/Results

In the following subsections we present the experimental results for four sets of servers. In the first subsection the configuration of the servers and loads that have to be processed are addressed. After that, the results are presented.

4.1 Configurations

We consider four configurations consisting of 16 nodes each. Every node can be a single server or a cluster of identical servers for which a single power function exists.

Configuration I consists of eleven single systems and five clusters. There are two 2-server clusters, two 4-server clusters and one 38-server cluster. The SPEC performance to power ratio is between $1,588$ and $3,038$ ssj_ops/Watt. Power consumption is at about 3.2 to 10.5 kW. The server hardware got available between September 2009 and June 2010. All servers are equipped with Intel Xeon processors (7 x X5670, 5 x L5530, 3 x L5520, 1 x E5540). Main memory is 4 GB (2 servers), 8 GB (38 servers) or 12 GB (4 servers). Each server is equipped with one or two processors containing four or six cores. All systems running Windows Server 2008, 15 of them Enterprise Server, one Datacenter Edition. The best performance to power ratio was achieved by a two-node multi-system, Intel Xeon 5670 at 2.9 GHz running Windows 2008 Datacenter Edition. The worst performance to power ratio was achieved by a two-node multi-system, Intel Xeon L5520 at 2.0 GHz running Windows 2008 Enterprise Server.

Configuration II consists of 14 single systems (three duplicates) and two 4-server clusters (identical systems). The SPEC performance to power ratio is between $3,775$ and $5,887$ ssj_ops/Watt. Power consumption is at about 1.4 to 5.8 kW. The server hardware got available between May and December 2012. One server is equipped with an AMD processor (Opteron 6380), the others with Intel processors (4 x E5-2470, 8 x E5-2660, 1 x E3-1265LV2, 2 x E5-4650L). Main memory is 8 GB (1 server), 16 GB (8 servers), 24 GB (8 servers), 32 GB (2 servers), 48 GB (2 servers) or 64 GB (1 server). Each server is equipped with one, two or four processors containing four, eight or 16 cores. All systems running Windows Server 2008 Enterprise Edition. The best performance to power ratio was achieved by an Intel Xeon E3-1265VL2 system at 2.5 GHz. The worst performance to power ratio was achieved by an Intel Xeon E5-2470 at 2.3 GHz.

Configuration III consists of 16 single systems. The performance to power ratio is between 245 and $1,746$ ssj_ops/Watt. Power consumption is at about 2.6 to 4.6 kW. The server hardware got available between November 2007 and April 2009. Two servers are equipped with AMD Opteron processors (2356, 8376HE), all others with Intel Xeon processors (1 x 5160, 2 x E5420, 1 x E5440, 1 x E5462, 1 x E5472, 2 x L3360, 1 x L5420, 2 x L5430, 2 x L5570, 1 x L7345). Main memory is 4 GB (2 servers), 8 GB (4 servers), 12 GB (1 server), 16 GB (7 servers), 18 GB (1 server) or 32 GB (1 server). Each server is equipped with one, two, four or eight processors containing two or four cores. Seven systems running Windows 2003 Server Enterprise Edition, four servers Windows 2008 Enterprise Server, two SuSE Linux ES 10, one Red Hat EL 5.3 and two Mac OS X 10.5.6 Server. The best performance to power ratio was achieved by an Intel Xeon 5570 system at 2.9 GHz running Windows 2008 Server. The worst performance to power ratio was achieved by an Intel Xeon E5472 system at 3.0 GHz running Mac OS X Server 10.5.6.

Interesting are the results of two systems of configuration III which are both equipped with an Intel Xeon 5570 processor running at 2.9 GHz (referred as system A and system B). The main differences are memory, hard disk and operating system/Java Virtual Machine. While system A is equipped with 12 GB (6x2 GB, PC3-8500E), system B is equipped with 18 GB (6x2 GB, 6x1 GB, DDR3 ECC FB-DIMM, 1066 MHz). The hard disk is a 2.5 inch 50 GB SSD (system A), a normal 160 GB HDD (system B), respectively. System A is running Windows 2008 Enterprise x64 with Oracle JRockit 6 while system B is running Mac OS X server 10.5.6 with Sun Java HotSpot 1.6.0 VM. Although both systems are similar in many respects, system A has with $1,746$ ssj_ops/Watt a nearly four times higher performance to power ratio than system B with 464 ssj_ops/Watt. The idle power of system B is about 2.3 times higher than that of system A. Both systems got available in early 2009 and both benchmark runs took place in March 2009.

Configuration IV consists of 12 single systems and 4 clusters. There is a 4-server cluster, a 16-server cluster, a 18-server cluster and a 32-server cluster. The SPEC performance to power ratio is between 268 and $5,521$ ssj_ops/Watt. Power consumption is at about 5.3 to 19.1 kW. The server hardware got available between November 2006 and October 2012. Two systems are equipped with AMD Opteron processors (2356 and 8376HE), 14 systems with Intel Xeon processors (5160, E5420, E5440, E5462, E5472, 2 x L3360, 2 x L5420, 2 x L5430, L7345, 2 x X5570). Main memory is 4 GB (2 servers), 8 GB (34 servers), 12 GB (4 servers), 16 GB (3 servers), 24 GB (34 servers), 32 GB (3 servers) or 128 GB (2 servers). Each server is equipped with one, two or eight processors con-

taining two, four, six or eight cores. Eight systems running Windows 2008 Enterprise Edition, six systems Windows 2003 Enterprise Server, one system SuSE Linux ES 10, one system Red Hat EL 5.3. The best performance to power ratio was achieved by an Intel Xeon E5-2470 16-node cluster at 2.3 GHz running Windows 2008. The worst performance to power ratio was achieved by an Intel Xeon 3040 system at 1.8 GHz running Windows 2003.

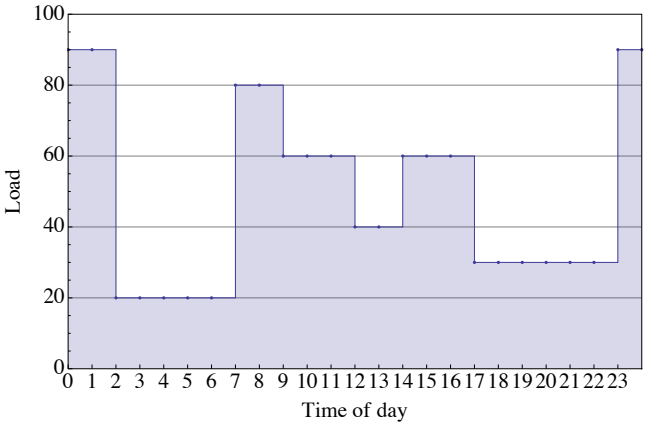

Figure 6. Load over day for an example server configuration

For energy evaluation we used a typical load profile of a server farm over a day providing a cluster of data base servers, application servers and dedicated login servers – all systems are virtualized. We simplified the load levels at each hour to full tenth of percent of load, see Fig. 6. Between 2300 and 0200 hrs some data base jobs are executed each night leading to a load of 90%. Between 0200 and 0700 hrs the servers are at 20% load. Between 0700 and 0900 hrs a significant amount of employees log into the system leading to a load at 80%. At working hours between 0900 and 1200 hrs as well as 1400 and 1700 hrs the load is at 60% while at lunch time between 1200 and 1400 hrs the load is at 40%

4.2 Results

The results for power reduction using our algorithms are described for all four configurations introduced in the last subsection. Above that, the energy consumption for the example load over day is given. Finally, we have a look on the power saving potential of the power-off algorithm which we applied on one of the configurations.

Fig. 7, 8, 9 and 10 show the results of all four configurations. For all configurations *bppf* or *ald* or a combination of those led to the lowest power consumption. The results of *ald* are quite good in all cases whereas *bppf* led to the worst overall result in configuration II. The power consumption at 0% and 100% load is, as expected, the same for all strategies. Analyzing the results we focus on the scope between 10% and 90% load. As *rlb* is a common strategy for load balancing we compare the other strategies with it regarding improvement potential.

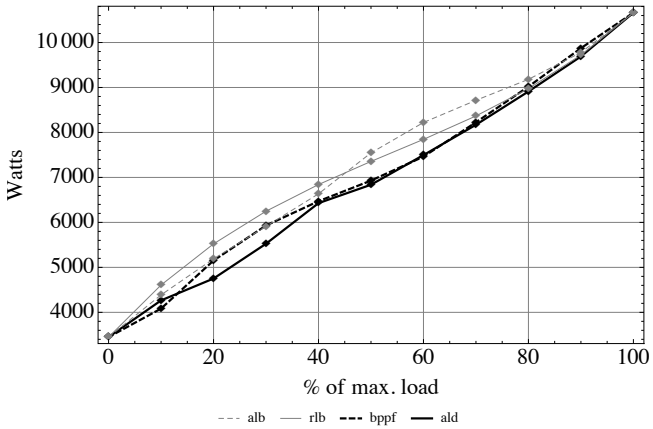

Figure 7. Results based on configuration I

In **configuration I**, see Fig. 7, *bppf* has the best result at 10% and 60% load, in all other cases *ald* outperforms the other strategies. *rlb* starts weak but has better results at load levels above 50%. *alb* is in the mid-field up to 50% load and in one case at 30% also better than *bppf* but is weak at higher load levels up to 80%. The power consumption with *ald* is up to 14% (at 20% load) lower than that of *rlb*. In the average *ald* leads to about 6% lower power consumption. The power consumption of *bppf* is in two cases (80% and 90% load) higher than *rlb* and in the average about 4.3% lower than *rlb*. At higher load levels the differences between the strategies are fading. The highest improvements are gained up to 50%. At 80% and 90% the difference between *rlb* and the other strategies is below 1%.

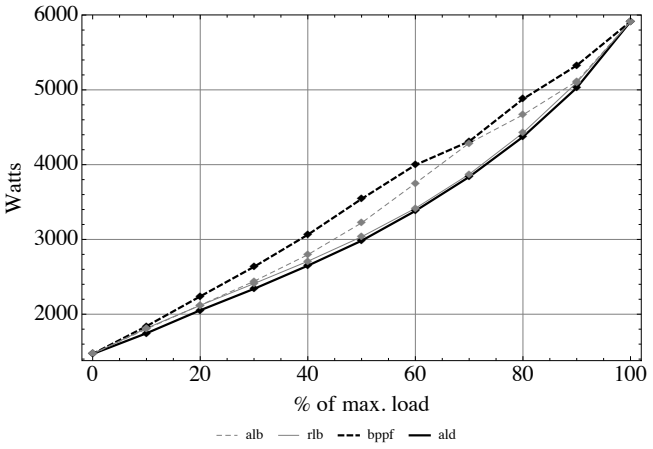

Figure 8. Results based on configuration II

In **configuration II**, see Fig. 8, *bppf* leads to the highest power consumption at all load levels while *ald* has the lowest power consumption. The results of *rlb* are quite good in this configuration. A similar curve with slightly worse results is achieved compared to *ald*. The results of

alb are in the mid-field. *ald* led to 3.7% less power consumption than *rlb* (at 10% load). The average is at about 2% lower power consumption. As all systems in this configuration got available between May and December 2012 and all running the same operating system we can suppose that the hardware power management features are somewhat similar. In that cases a normal load balancing scheme would lead to acceptable power consumption compared to the *ald* strategy.

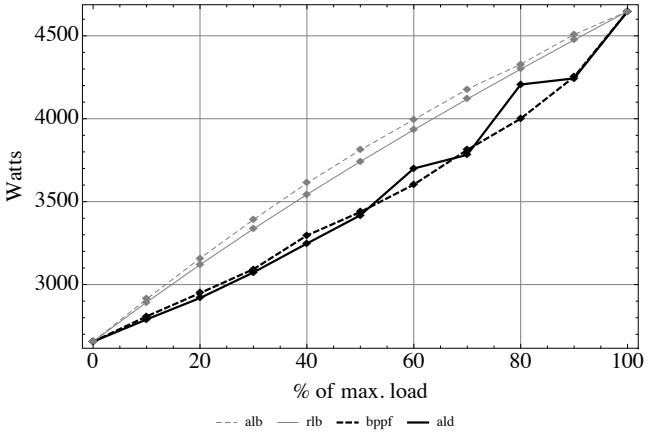

Figure 9. Results based on configuration III

In **configuration III**, see Fig. 9, *bppf* performs well, despite the fact that the results are a bit worse than those of *ald* in most cases. But *ald*, using different strategies to minimize the α-values for work assignment, has problems at 60% and 80% load, where the power consumption is significantly higher than achieved by the more stable *bppf*. The rounds to minimize the α-values are limited for *ald*. More rounds lead to better results at the cost of run-time – the run-time is orders of magnitude greater than the run-times of the other strategies. Both, *alb* and *rlb* have similar results with *rlb* a little bit better but much worse than *bppf* or *ald*. *ald* led to 8.7% less power consumption than *rlb* (at 50% load). The average is about 6.3% (*ald*) and 6.5% (*bppf*).

In **configuration IV**, see Fig. 10, *ald* performs well. Only at 10% load *bppf* leads to a much lower power consumption, at 50% and 60% to a slightly lower power consumption. On higher load levels over 80% *bppf* has the worst results. Both, *rlb* and *alb* are worse than the other two strategies up to 70% load. *ald* led to 18.2% less power consumption than *rlb* at 30% load. The average is at about 8.9%. This configuration covers systems gotten available over a long period of time from November 2006 to October 2012. In these cases, where different hardware with different power reduction facilities is used, the most advantages compared to a simple load balancing (*rlb*) can be achieved. The greatest benefit is achieved at lower load levels up to 50%.

In all four configurations either *ald* or *bppf* lead to the lowest power consumption.

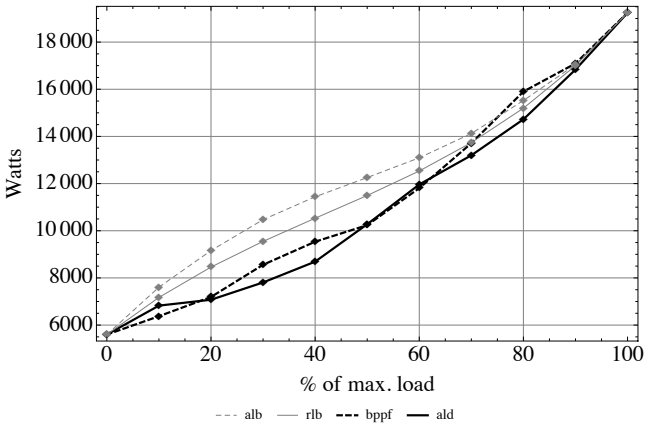

Figure 10. Results based on configuration IV

Table 2 shows the energy consumption over a day according to the load levels in Fig. 6 for all four configurations. The lowest result is written bold. The last row shows the quotient $best/rlb$, where best is the result of the strategy with the lowest energy consumption.

Conf.	alb	rlb	bppf	ald	$best/rlb$
I	653	657	635	**616**	0.94
II	300	289	317	**283**	0.98
III	338	334	**311**	314	0.93
IV	1,113	1,059	999	**965**	0.91

Table 2. Energy consumption over day for all configurations in MJ

Regarding the results in power consumption, *ald* and *bppf* lead to the best results. The run-times of the strategies for all configurations are depicted in Tab. 3. *ald* is four to six orders of magnitude slower than the other strategies for configurations with 16 servers. If increasing the number of servers it becomes necessary to increase the rounds to calculate usable results in the case of *ald*. This increases the run-time of *ald* significantly, while the run-times of the other algorithms increase much slower. If the strategies are used off-line to calculate distributions for distinct load levels once in advance as we intended, this is not a great problem. Using *ald* on-line is not recommended. *rlb* is the fastest strategy, two orders of magnitude faster than *alb*

Conf.	alb	rlb	bppf	ald
I	0.00867	0.00004	0.00341	55.73920
II	0.00606	0.00005	0.00407	26.78819
III	0.00649	0.00005	0.00373	31.89179
IV	0.00338	0.00006	0.00338	58.36603

Table 3. Run-times of the strategies in seconds

A huge amount of power and energy can be saved by switching off servers due to their high idle power consump-

tion, particularly if the overall load is low (up to 70%). At higher load levels the potential decreases more and more as less machines can be switched off if any at all. Especially, *bppf* is eligible for powering-off unused machines. As only the first k of n machines ($k \leq n$) are designated to do work, the remaining systems can be shut down. For *ald* we presented an algorithm to decide which machines to switch off. The whole procedure can take place off-line, so there is no real need to redistribute work on-line for the purpose of identifying the machines to switch off. Provisioning plans are calculated and applied when a specific load level is reached. In that case all work is distributed as calculated in advance.

Because *rlb* and *alb* distribute the workload somewhat equally over all machines, the results of these strategies are less suited for a power-off scheme.

5 Conclusion

We presented methods for calculating the power-optimized workload assignment to a collection of heterogeneous servers. These methods can be further refined for scheduling strategies of non-divisible loads to find optimized solutions regarding power consumption in large computing environments. With information about future performance needs rules can be devised when to move specific load packages from one server to another to balance workload in a power-efficient manner.

The iterative approach also considers the possibility to switch off one or more servers to further reduce power consumption. With known performance demands it is possible to turn off servers e.g. at night time when less performance is needed. Intelligent power management facilities could be established to redistribute loads at specific times, turn servers on and off and collect data about performance demands to adapt performance information to changing situations.

In follow-up work we will investigate genetic algorithms to find good solutions for the selection of servers to switch off. An initial set of chromosomes consisting of randomized solutions and solutions calculated by our iterative approach are genetically manipulated over several rounds using mutation and crossover operations. In an evaluation process the fitness values are calculated using the power functions and minimized values of α.

Notes

References

[1] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*, 37(11):68–76, November 2004.

[2] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, pages 67–77, 2011.

[3] P. Cichowski, J. Keller, and C. Kessler. Energy-efficient mapping of task collections onto manycore processors. In *5th Swedish Workshop on Multicore Computing*. MCC, 2012.

[4] J. Cong and B. Yuan. Energy-efficient scheduling on heterogeneous multi-core architectures. In *ACM/IEEE Int. Symp. on Low Power Electronics and Design*, pages 345–350. ACM, 2012.

[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. *INRIA*, 2010.

[6] E. Feller, C. Rohr, D. Margery, and C. Morin. Energy management in IaaS clouds: A holistic approach. In *5th Int. Conf. Cloud Computing*, pages 204–212. IEEE, 2012.

[7] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 5th edition, 2011.

[8] J. Howard et al. A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. In *IEEE Int. Solid-State Circuits Conf. Digest of Tech. Papers*, pages 108–109, 2010.

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

[10] Cong L. Energy-efficient workload mapping in heterogeneous systems with multiple types of resources. In *21st Int. Conf. on Parallel Architectures and Compilation Techniques*, pages 491–492. ACM, 2012.

[11] E. Pinheiro and R. Bianchini. Load balancing and unbalancing for power and performance in cluster-based systems. *Workshop on Compilers and Operating Systems for Low Power*, 2001.

[12] 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.

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

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

[15] I. Takouna, W. Dawoud, and C. Meinel. Energy efficient scheduling of HPC-jobs on virtualize clusters using host and VM dynamic configuration. *SIGOPS Oper. Syst. Rev.*, pages 19–27, 2012.

[16] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. In *Middleware*, volume 5346 of *Lecture Notes in Computer Science*, pages 243–264. Springer, 2008.

[17] G. von Laszewski, L. Wang, A.J. Younge, and X. He. Power-aware scheduling of virtual machines in DVFS-enabled clusters. In *IEEE Int. Conf. on Cluster Computing and Workshops*, pages 1 –10. IEEE, 2009.

[18] P. Weisberg and Y. Wiseman. Efficient memory control for avionics and embedded systems. *Int. J. of Embedded Systems*, 5(4), 2013.