Extending the Cutting Stock Problem for Consolidating Services with Stochastic Workloads

Markus Hähnel, John Martinovic, Guntram Scheithauer, Andreas Fischer, Alexander Schill, Waltenegus Dargie, *Senior Member, IEEE*

***Abstract*—Data centres and similar server clusters consume a large amount of energy. However, not all consumed energy produces useful work. Servers consume a disproportional amount of energy when they are idle, underutilised, or overloaded. The effect of these conditions can be minimised by attempting to balance the demand for and the supply of resources through a careful prediction of future workloads and their efficient consolidation. In this paper we extend the cutting stock problem for consolidating workloads having stochastic characteristics. Hence, we employ the aggregate probability density function of co-located and simultaneously executing services to establish valid patterns. A valid pattern is one yielding an overall resource utilisation below a set threshold. We tested the scope and usefulness of our approach on a 16-core server with 29 different benchmarks. The workloads of these benchmarks have been generated based on the CPU utilisation traces of 100 real-world virtual machines which we obtained from a Google data centre hosting more than 32 000 virtual machines. Altogether, we considered 600 different consolidation scenarios during our experiment. We compared the performance of our approach – system overload probability, job completion time, and energy consumption – with four existing/proposed scheduling strategies. In each category, our approach incurred a modest penalty with respect to the best performing approach in that category, but overall resulted in a remarkable performance clearly demonstrating its capacity to achieve the best trade-off between resource consumption and performance.**

***Index Terms*—Bin-packing problem, cloud computing, consolidation, cutting stock problem, data center, energy-efficient computing, workload consolidation, server consolidation.**

I. Introduction

As cloud computing takes roots within the fabrics of the Internet, a good portion of the IP traffic will be processed and stored in data centres. According to Cisco Systems, the global data centre IP traffic has been exhibiting a rapid and sustained rise in the past several years. For example, in 2017 the amount of cloud data centre IP traffic was 8190 exabytes for the year and it is expected to increase to 19 509 exabytes per year by 2021 [1]. This inevitably requires the deployment of more capable processing and storage servers in the coming years. On the other hand, however, independent studies suggest that existing servers are not utilised optimally, as platform providers tend to over-provide resources in order to guarantee high availability at peak load [2]–[4].

The magnitude of data centre workload for different applications fluctuates significantly as a function of time [5]–[7]. For some applications, nevertheless, the pattern appears to be persistent or even deterministic over several years. For others, sufficient statistics can be gathered in a short time, so that accurate predictions can be made on hourly basis or even on shorter time scales. This aspect can be employed to balance the demand for and the supply of computing resources in data centres and large-scale server clusters (in other words, both underutilisation and overloading of servers can be avoided without violating a Service Level Agreement during peak load).

Fig. 1 displays a recent statistics we obtained from Google Transparency Report[1]. The statistics refer to the normalised workload handled by the Google Search application in Ger-

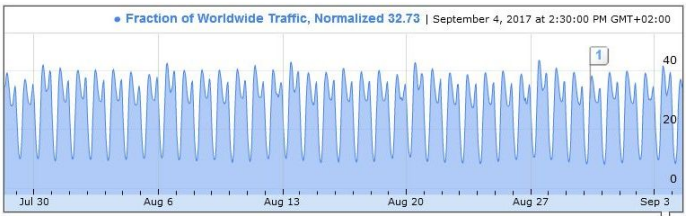


Fig. 1: The normalised traffic of Google Search for Germany. The normalised traffic is obtained by computing the ratio of Germany's request rate to the worldwide request rate, resulting in a number between 0 and 1. Then Google multiplies the number by a constant, which normalises but does not change the shape of the graph. The same approach was adopted in Fig. 2.

[1]https://www.google.com/transparencyreport/traffic/data/ (Last visited on September 12, 2017, 11:45 AM CET)

Fig. 2: The normalised traffic of YouTube for Germany.

Revised manuscript was received on 19 March 2018.
Revised manuscript was received on 26 January 2018.
Manuscript was first received on 1 October 2017.

M. Hähnel, A. Schill and W. Dargie are with the Technische Universität Dresden, Chair of Computer Networks, 01062, Dresden, Germany.
E-mail: {markus.haehnel1, alexander.schill, waltenegus.dargie}@tu-dresden.de

J. Martinovic, G. Scheithauer, and A. Fischer are with Technische Universität Dresden, Institute of Numerical Mathematics, 01062, Dresden, Germany.
E-mail: {john.martinovic, guntram.scheithauer, andreas.fischer}@tu-dresden.de

This work has been partially funded by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-efficient Computing" (HAEC).

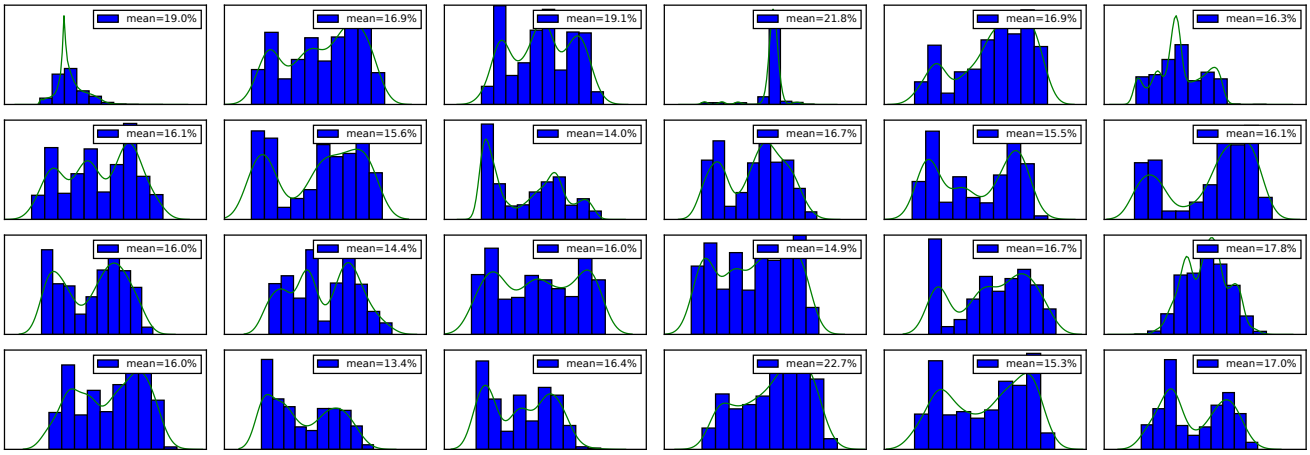


Fig. 3: A histogram summary of the CPU utilisation of 24 virtual machines from the Google data trace[2].

many between June 30 and September 3, 2017. As can be easily seen, at a micro scale, the data centre workload behaved as a random process whereas at a macro scale, the pattern is almost deterministic. Similarly, Fig. 2 shows the traffic pattern of YouTube Germany from April 30 to May 6, 2013. Here, too, one sees a significant short term fluctuation, but with a macro scale statistics which can be regarded as a wide-sense stationary stochastic process. It is also worth noting that the workloads of the two applications have different statistics as well as resource utilisation characteristics. Fig. 3 summarises the resource utilisation statistics (histogram) of 24 of the 32 000 virtual machines which we have analysed for this study. The virtual machines ran on 12 500 physical servers on a Google data centre[2] for a period slightly exceeding a month in May 2011. As can be seen in the figure, the resource (CPU) utilisation fluctuated over time clearly suggesting that it has to be regarded as a random process or a random variable.

We assert that the short and long term statistics of co-located services can be employed to consolidate servers at runtime, so as to achieve high energy efficiency in data centres. The short-term behaviour enables to model the workload of a service as a random variable whilst the long-term pattern can be used to determine the probability density function of the workload. Furthermore, the convolution of the density functions of multiple services provides insight into the resource utilisation characteristics when these services are scheduled to execute on one and the same platform. In other words, the outcome of a convolution is sufficient to predict the extent of overload during service consolidation.

In this paper we propose a service consolidation strategy by extending the Cutting Stock Problem (bin-packing problem) for stochastic workloads. Originally proposed for the economical use of material at the cutting stage [8], the strategy has found a wide range of applications dealing with assortment problems. The prevailing idea is that given a set S of n services, to generate $m \leq n$ disjoint subsets, so that the services in a subset can be executed simultaneously on one and the same platform without actually exceeding the maximum computing capacity of the platform.

[2]https://github.com/google/cluster-data

We summarise the contribution of this paper as follows:

- To the best of our knowledge ours is the first to apply the cutting stock problem for stochastic, arbitrary workloads.
- We determine, both analytically and experimentally, the conditions in which a deterministic and a probabilistic cutting stock problem (CSP) yield comparable performance.
- We verify the scope and usefulness of our approach with practical experiments. By employing actual resource utilisation traces of more than 32 thousand virtual machines from a Google data centre to generate realistic workload for 29 different benchmarks, we compare the performance of our approach with the performance of four baseline schedulers: the Linux scheduler, a random scheduler, a scheduler proposed in [9], and a deterministic CSP scheduler, by defining different performance and execution metrics.

The rest of the paper is organised as follow: In Section II, we discuss related work and in Section III, we put in place the background of our work. Section IV describes the extended CSP which is proposed to consolidate services having stochastic workloads. Section V describes the experimental set-up for the evaluation of our approach. The evaluation itself is given in Section VI. Finally, Section VII gives concluding remarks.

II. Related work

Tailoring the available computing resources (such as the number of CPUs) to the amount of anticipated workloads in data centres and large-scale server clusters to manage a large number of virtual machines (VMs) is an ongoing research area [10]–[13].

Yi *et al.* [14] propose a new cloud-based computing paradigm named *Cocoa*. The basic idea is to organise different jobs with complementary resource demands into groups so that they can be allocated to group buying deals as pre-defined by cloud providers, in order to be executed in virtualised containers accordingly. The major challenge, of course, is to identify appropriate job groups based on their expected resource demands. In this paper, the initial static group organisation is modelled as a variable-sized vector bin-packing

problem, whilst the subsequent dynamic group organisation problem is represented as an online multidimensional knapsack problem. The effective combination and interaction of both phases enables significant performance improvements during workload consolidation. This is confirmed by the detailed simulation-based evaluation that considers the different strategies, compares them with existing approaches, and also provides interesting different viewpoints, for example, from the perspective of the end user versus the perspective of the cloud provider. In our work, we employ a different optimisation approach that can be applied in real time more readily, with the goal to validate it based on a real implementation rather than a simulation.

Yu *et al.* [15] define a probabilistic threshold of exceeding the capacity of a server in a cloud environment hosting a large number of VMs. If this threshold is crossed, the server is labelled as a "hotspot". Subsequently, they order the hotspots by decreasing overload risk. Then, the authors go through each hotspot, starting with the server with the highest overload risk. The VMs on each server are sorted as well as grouped by a decreasing metric describing the potential of reducing the server's overload risk. Finally, the VMs from the group with the highest potential but lowest migration cost are migrated until the server is no longer a hotspot. Similar to our approach, the authors adopt assigning a "capacity exceeding probability" in their algorithm. Furthermore, they propose a modified First Fit Decreasing (FFD) algorithm in order to load balance the workload from the hotspots. However, they focus only on overloaded servers as a result of which the algorithm optimises only locally, whereas we strive to find a global near-optimum solution. Additionally, we investigate the influence of the probability threshold on the power consumption as well as the performance.

Beloglazov and Buyya [16] consider VM consolidation consisting of three subtasks: (1) the detection of overloaded and underutilised servers, (2) the selection of candidate VMs for migration, and (3) the identification of target servers to which VMs can be migrated. As heuristic for overloaded and underutilised servers they use thresholds based on the statistical analysis of historical, real workload traces of the CPU utilisation. Thus, in (1), the threshold of overload situation is adaptively defined by the median of the absolute deviations from the median of the data set. For underutilised situations the difference of the first and third quartiles of CPU utilisation is considered. In (2), the future CPU utilisation is estimated by the previous observation via a local regression method. For identifying VMs for migration they investigate three algorithms. In the first, the VMs with the shortest migration time are chosen. In the second, a high correlation of the CPU utilisation signalises the source VMs. The third algorithm chooses VMs randomly. Finally in (3), the VM consolidation is described as a CSP with CPU utilisation as item length and power consumption as costs. All three subtasks are evaluated using the CloudSim simulation framework[3] considering more then 1000 VMs having only a single core assigned to each. The authors report that the combination of the regression and the minimum migration time approaches produces the best results when both energy saving and minimisation of Service Level Agreement (SLA) violation are considered at the same time. The combination of the fixed threshold overloading heuristic with the minimum migration time produces the highest energy saving but at the price of higher SLA violations. The experiment of Beloglazov and Buyya [16] is based on real workload traces including the fluctuations of the VM. The evaluation itself is, nevertheless, performed in a simulator, whereas we validate our concept by real measurements. Moreover, our approach endeavours to assign as many jobs as possible to a single core in order to avoid underutilisation at a core level whilst here a single job is assigned to a core.

In contrast to [16], Verma *et al.* [9] assume that there already exists a mechanism for predicting the optimal resource consumption of a VM to meet its SLA. Furthermore, they demand that it is possible to order all servers monotonously with respect to a global power-efficiency for all services. Then they order the servers by decreasing energy-efficiency and the VMs by decreasing optimum resource consumption. Subsequently, they assign the VMs from the list to the most energy-efficient server until its capacity is reached. In the next iteration they assign the remaining VMs from the list to the second most energy-efficient server and so on until all VMs are mapped to a server. Indeed, Verma *et al.* [9] covers heterogeneous computing resources (e. g. physical machines (PMs) or cores) with individual power profiles but assume static resource consumption. Thus, workload with random resource consumption, as we address in this paper, exceeds the resource capacity quite often. We make an experimental comparison with this approach in Section VI.

Xu *et al.* [17] consider 50 single-core as well as quad-core VMs on 10 PMs while minimising energy consumption and avoiding SLA violations. For the latter they include interference aspects on CPU, network I/O, CPU cache and memory bandwidth described by a multi-dimensional supply-demand model. The approach was tested with several static workloads such as SPEC CPU2006[4] and netperf[5]. CPU and memory intensive workloads could be improved by 16 to 28 % and network-intensive workloads by 45 to 65 % in comparison to the strategies in [9]. Even though the proposed approach is self-sufficient, its interference aspect can be integrated into existing load balancing strategies.

Eichhorn *et al.* [18] investigate the power consumption, throughput and latency of a distributed video streaming server. They compare four different strategies for workload consolidation. First, all nodes are turned on without performing any migration or load-balancing as reference. Second, still all nodes are powered up but throughput and latency is improved by distributing popular videos for optimised network usage. Third, only nodes are powered on which host currently or recently demanded videos. Fourth, always powered on nodes host popular videos, and nodes with predominantly unpopular videos are turned on *on demand* while the load of all active nodes is balanced. None of these strategies yields an optimum

[3]https://code.google.com/p/cloudsim/

[4]https://www.spec.org/cpu2006/

[5]https://hewlettpackard.github.io/netperf/

solution for power consumption, throughput and latency at the same time. However, the fourth strategy, with the selection of the appropriate threshold for popularity, can achieve most of these goals. While this approach yields good results for video platforms thanks to the prediction techniques employed, these techniques, nevertheless, are difficult to reuse for arbitrary workloads.

Homsi *et al.* [19] take a similar approach as we do. Here, the expected performance is expressed as an M/M/1 queuing problem and the service-to-VM assignment uses the FFD algorithm. However, their approach does not admit arbitrary workloads because the characteristic of each service must be one of a predefined set. Furthermore, a service rate determines the resource consumption. Thus, there is no black-box scheduling based on generic parameters such as the CPU utilisation possible. By contrast, our approach admits any workload as well as computing resource. Arroba *et al.* [20] also use the FFD algorithm in combination with Dynamic Voltage and Frequency Scaling (DVFS) [21]. Thus, their solution always prefers the lowest feasible frequency for each server, since the power consumption of the server is proportional to the frequency of the CPU. For our case, we disabled DVFS in order to quantify the merits and demerits of service consolidation in isolation.

In summary, the proposed approaches are tested with either simulation environment, scaled down hardware, or static benchmarks. Often, the power and energy consumptions are estimated instead of measured. By contrast, our approach includes the dynamic fluctuations of resource consumption and is performed on a real multi-processor server. In particular, the inclusion of the resource consumption fluctuations in our model enables a lower consolidation frequency because it already considers the aspects which can change at runtime.

III. Background

Before we delve into the description of our concept, we shall discuss some preliminaries concerning the FFD algorithm and random variables in order to establish a shared understanding.

A. Consolidation with First Fit Decreasing

The prevailing idea is that given a set S of n services, to generate $m \leq n$ disjoint subsets $S_1, \dots S_m$, so that the services in a subset can be executed simultaneously on one and the same computing platform without actually exceeding the maximum computing capacity of the platform. As a basic example, consider Fig. 4. Suppose we have n services each having a static resource demand (workload) expressed by μ_i. We wish to determine the optimal amount of computation resource for handling the overall workload generated by the n services. We describe a subset S_i of services, named a *pattern*, as a binary vector of n elements corresponding to the indices of services contained in a subset – for example, for the first two subsets in Fig. 4, we have:

$$\vec{a}_1 = (1, 0, 1, \dots)^\intercal, \quad \vec{a}_2 = (0, 1, 0, 1, 1, \dots)^\intercal. \tag{1}$$

A *feasible pattern* $\vec{a}_i$ is defined by a subset S_i of services the collective resource consumption of which does not exceed

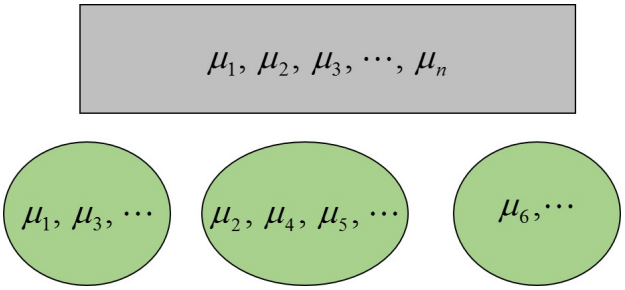


Fig. 4: Given a set S of n services, each service having a resource demand of μ_i, the CSP aims to generate m disjoint subsets, so that the services in a subset form a *pattern* and can be executed simultaneously on one and the same platform.

the available capacity of the node on which they are executed simultaneously. If each unit computing node (for example, a single processor core or a unit server) has a maximum capacity of C and the workload μ_i requires a portion of C, then the overall resource consumption of the entire subset should not exceed C or should exceed only by a margin foreseen in an SLA. In this sense, a *feasibility criterion* describes whether a pattern is feasible or not:

$$\sum_{k \in S_i} \mu_k \leq C. \tag{2}$$

Hence, the set A of all feasible patterns for a consolidation assignment can be described by

$$A = \left\{ a \in \mathbb{B}^n | \vec{a}^\intercal \vec{\mu} \leq C \right\} \tag{3}$$

where $\vec{\mu} = (\mu_1, \mu_2, \dots, \mu_n)^\intercal$ is a vector containing the resource demands of all the services to be consolidated.[6] In order to ensure that the computing node is not overloaded during consolidation, the capacity C can be replaced by an effective capacity $c < C$.

Solving the CSP for obtaining an exact, optimal solution is $\mathcal{NP}$-hard [22]. Nevertheless, there are models and algorithms which can yield the optimal solution even for a large number of services, but they require integral input parameters [23]–[25]. Unfortunately, scaling real numbers to integral values is not simple and often leads to large numbers. Hence, for an efficient and fast service consolidation, the most feasible approach is employing an approximation of the optimal solution. One of these is the FFD algorithm, which yields a near optimal solution [26], [27].

It is carried out in three steps:

1) Sort all services in S according to their mean resource utilisation in decreasing order.
2) Initialise an empty pattern $\vec{a}_1$.
3) For each service $s_i \in S$, find the lowest-indexed pattern $\vec{a}_k$, such that service s_i can be added to $\vec{a}_k$ without violating a feasibility condition. If such a pattern does not exist, generate a new (empty) pattern and assign service s_i to it.

The FFD first sorts the services according to their expected resource utilisation, in a decreasing order. As a second step, it

[6] The CSP is solvable if and only if $\mu_i \leq C, \forall i$. If this is not the case, then there is no point assigning this task to a single computing node.

creates the first subset (pattern $\vec{a}_1 = (0, \ldots, 0)^\intercal$) and assigns the first service to it. If the resource demand of the second service still fits into the first subset, it assigns the second service to the first subset, otherwise, it creates a new subset (pattern $\vec{a}_2$). If the resource demand of the third service fits into the first subset, it assigns the third service to the first subset, otherwise, it tries the second subset. If both subsets cannot accommodate the third service, it creates a new subset (pattern $\vec{a}_3$), and so on. The detailed implementation of the FFD algorithm including the assignment of patterns to a computing node is given in Algorithm 1.

Algorithm 1 CSP by First Fit Decreasing (FFD)

Input: set of all services S

1: Sort all services S according to non-increasing mean values, i. e. $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$.
2: Initialise an empty pattern $\vec{a}_1$.
3: **for all** service $s_i \in S$ **do**
4: **for** $k = 1$ **to** number of current initialised patterns **do**
5: **if** $<$ *feasibility criterion* $>$ **then**
6: Add i to pattern $\vec{a}_k$.
7: Assign service s_i to core k.
8: **break**
9: **end if**
10: **end for**
11: **if** service s_i is not assigned yet **then**
12: Initialise a new empty pattern $\vec{a}_{k+1}$.
13: Add i to pattern $\vec{a}_{k+1}$.
14: Assign service s_i to core $k + 1$.
15: **end if**
16: **end for**

B. Workload as random variable

Since the resource utilisation of a service at any instance of time may not be known except in a probabilistic sense, we model services by the statistics of their resource consumption characteristics. In other words, we regard them as random variables. We use boldface (for example, $\mathbf{u}$) to express a random variable. The probability that the utilisation of the random variable $\mathbf{u}$ is below or equal to a certain real value u is expressed by the cumulative distribution function: $F(u) = P\{\mathbf{u} \leq u\} \geq 0$. Likewise, the probability that the random variable $\mathbf{u}$ will have a value between $(u, u + \mathrm{d}u)$ is given by the probability density function (pdf): $f(u) = \mathrm{d}/\mathrm{d}u\ F(u)$. If we have either the distribution or the density function, then we have sufficient information to characterise a random variable. Two of the properties of a random variable which can be determined by either of these functions and have great statistical significance are the mean (μ) and the variance (σ^2). The variance, in particular, expresses the uncertainty, for our context, about the resource utilisation of the service. The bigger the variance, the bigger is the uncertainty.

If we schedule two or more services to share a single computing node (for example, a processor core), their overall resource utilisation should likewise be characterised as a random variable. To demonstrate this: Suppose the resource

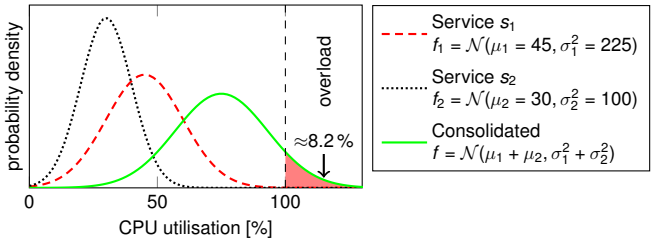


Fig. 5: The execution of two services with normally distributed resource utilisation characteristics. The shaded area demonstrates the probability of exceeding the maximum capacity of the processor.

utilisation characteristics of service s_1 and s_2 can be described by $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. Suppose also at any instance of time service s_1 utilises 12 percent of the CPU cycles with a probability of 0.2 and 30 percent with a probability of 0.8. Similarly, service s_2 utilises 10 percent of the CPU cycles with a probability of 0.6 and 50 percent with a probability of 0.4. Now if we observe the CPU load for 1 second, how does the CPU utilisation look like? Since the overall CPU utilisation is the summation of two random variables, we have: $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2$. This will yield four different outcomes with different probabilities: $12\,\% + 10\,\% = 22\,\%$ ($P = 0.12$), $12\,\% + 50\,\% = 62\,\%$ ($P = 0.08$), $30\,\% + 10\,\% = 40\,\%$ ($P = 0.48$), or $30\,\% + 50\,\% = 80\,\%$ ($P = 0.32$). Thus, the overall CPU utilisation has its own probability distribution. If the resource utilisation of the two services is continuous, the probability distribution of the overall CPU utilisation is also continuous and is determined by the convolution operation:

$$f(u) = \int f_1(\alpha) \cdot f_2(u - \alpha)\, \mathrm{d}\alpha \tag{4}$$

where $f(u)$ is the pdf of the overall CPU utilisation, $f_1(u)$ and $f_2(u)$ are the pdfs associated with the utilisations of the first and the second service, respectively.

Often, the convolution operation is computation intensive when a large number of services is involved. An exception to this is when the pdfs of the individual services assume Gaussian or normal distribution, even though each service may have a different mean and variance. In this case, the overall resource utilisation is also normally distributed with mean and variance as the summations of the means and variances of the k individual services[7]:

$$\mu = \sum_{i=1}^{k} \mu_i, \qquad \sigma^2 = \sum_{i=1}^{k} \sigma_i^2$$

Fig. 5 displays the pdf of the CPU utilisation when two services both having normal pdfs are simultaneously executed on one and the same processor. In the subsequent subsections we assume that the resource utilisation statistics of individual services (virtual machines) which we wish to consolidate can be approximated by the normal distribution. This is a common approach [29][30] and warrantable for the Google data trace [15].

[7]Note that $f(u)$ converges to be normal distribution even if the densities of the individual services are not normal. This is in accordance with the Central Limit Theorem [28].

IV. Extended First Fit Decreasing

The FFD algorithm from Section III-A takes neither the individual nor the aggregate variance into account when services are assigned to a computing node (i.e., to one of the disjoint subsets). As a result, a consolidation assignment may result in both underutilisation and overloading situations. In order to account for these conditions, we propose to examine the density of the overall resource utilisation and denote this variant of the CSP as cutting stock problem with nondeterministic item lengths (ND-CSP). The *feasibility criterion* during consolidation for this variant is expressed as:

$$P\left\{\mathbf{u} > C\right\} = \int_{C}^{\infty} f(u)\,\mathrm{d}u \leq \epsilon \tag{5}$$

where $\mathbf{u}$ is the overall resource utilisation of an entire subset modelled as a random variable, $f(u)$ is its density function, and ϵ is the maximum exceeding probability (MEP) which depends on the Quality of Service (QoS) that should be achieved[8]. The MEP describes the theoretical probability of exceeding the computing node's capacity within a tolerable limit. The bigger ϵ is the more services can feasibly be consolidated on one platform and, thus, the less computation resources are necessary. A higher overload probability, on the other hand, degrades the QoS. Hence, the MEP is a trade-off between reducing the amount of computing resources and the QoS in order to fulfil an SLA.

Fig. 5 demonstrates the *feasibility criterion* with the resource consumption characteristics of two normally distributed services. The consolidation of s_1 and s_2 is feasible as long as the MEP satisfies $\epsilon \gtrsim 8.2\,\%$.

With the maximum exceeding (overloading) probability (5) set in place, it is now possible to modify Equation (3) for the set A of all feasible patterns as follows:

$$A = \left\{a \in \mathbb{B}^n | P\left\{\vec{a}^\intercal \vec{u} > C\right\} \leq \epsilon\right\} \tag{6}$$

where $\vec{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n)^\intercal$ is a vector describing the collective resource demand of all services. As described in the section before, the overall resource utilisation of services simultaneously executing on one and the same node after a consolidation, i.e., $\mathbf{u} = \vec{a}^\intercal \vec{u}$, will be normally distributed with a pdf: $f(u) = \mathcal{N}(\vec{a}^\intercal \vec{\mu}, \vec{a}^\intercal \vec{\sigma}^2)$. The vectors $\vec{\mu} = (\mu_1, \ldots, \mu_n)^\intercal$ and $\vec{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_n^2)$ contain the means and variances, respectively, corresponding to the resource demand of all the services. However, Equation (5) still contains an integration which is often compute intensive. Instead, it is preferable to employ the quantile q_ϵ which can be defined by a random variable $\mathbf{x}$ with a pdf $f(x) = \mathcal{N}(0, 1)$:

$$P\left\{\mathbf{x} \leq q_\epsilon\right\} = 1 - \epsilon. \tag{7}$$

Note that the values of q_ϵ are tabulated for ordinary choices of ϵ ("error function"). If these values can be rounded down by accepting a reasonable amount of error, then the equality sign in (7) can be replaced by $\leq$, which still yields feasible patterns. Consequently, (6) can be simplified into:

$$A = \left\{a \in \mathbb{B}^n | \vec{a}^\intercal \vec{\mu} + q_\epsilon \cdot \sqrt{\vec{a}^\intercal \vec{\sigma}^2} \leq C\right\} \tag{8}$$

Thus, the *feasibility criterion* of potential candidates for a consolidation task can be determined by simply summing up the means and variances of these candidates.

[8]The ND-CSP is solvable if and only if $P\{\mathbf{u}_i > C\} \leq \epsilon, \forall i$. If this is not the case, then there is no point assigning this task to a single computing node without violating the *feasibility criterion*.

V. Experimental set-up

A. Benchmarks

We selected 29 benchmarks from the Mälardalen WCET research group [31] as candidate services for a consolidation assignment in a 16-core server. The consolidation assignment is determining the number of physical cores required in order to handle the workload of the services being consolidated, so that all idle cores can be put to a low power mode. Our choice of the benchmarks took into account the workload they generated and the diversity of their application logic. Nevertheless, the execution durations of these benchmarks were typically less than 1 ms. Therefore, we modified them, so that their execution durations can last several minutes and their resource consumption statistics can obey an underlying density function. In order to determine the resource utilisation characteristics (i.e., workload statistics), we relied on the execution traces of more than 32 000 real-world virtual machines which we obtained from a Google data centre[2]. The trace consisted of, among other things, the CPU utilisation of the virtual machines. Some exemplary CPU consumption characteristics are shown in Fig. 3. Using the traces, we computed the mean μ_i and variances σ_i^2 for all the virtual machines in order to select the first 100 traces which exhibited the highest variances in their CPU utilisation. Then we determined the execution patterns of the 100 virtual machines, so that their CPU utilisation reflected the statistics of the 29 benchmarks. Hence, with the 29 benchmarks and 100 workload statistics, we had altogether 2900 services to select for our experiment.

B. Measurement

Our experiment ran on a system with four sockets, each socket hosting an Intel Xeon E5-4603 quad-core. We disabled the HyperThreading in order to bind to physical cores only. The system had an active memory of 64 GB installed in NUMA-architecture and Ubuntu Server[9] (v. 16.04) as its operating system. With the 16 physical cores (each having a maximum capacity $C = 100\,\%$), we tested our extended algorithm to consolidate randomly selected $n = 5$, 10, 15, 20, 25 and 30 services having arbitrary execution characteristics (statistics). We compared our algorithm with four different scheduling algorithms. The first algorithm is the classical FFD which takes the mean resource utilisation of the consolidated services into account (Section III-A). This algorithm (which we label as meanCSP) was tested for different effective core capacities, namely, $c = 50$, 70 and 90 %. The second algorithm does a random service-to-core assignment where each service is pinned to one randomly chosen core. Hence, once assigned

[9]https://www.ubuntu.com/download/server

to a core, the services were not migrated between cores. The third algorithm is the one proposed by Verma *et al.* [9] (labelled as *Verma2008*) and discussed in detail in Section II. We implemented and fully integrated this algorithm with our system. Our choice was influenced by the completeness of the description of the algorithm for implementation. Finally, we employed the default Linux scheduler which aims to balance the load of cores by migrating services during runtime. For our algorithm (labelled as ND-CSP), we considered different MEPs, namely, $\epsilon = 0.05$, 0.1, 0.25 and 0.4. In order to avoid single best and worst case examples, we generated 10 instances for each consolidation. Altogether, we had 600 scenarios, each scenario (experiment) lasting 10 min. Algorithm 2 summarises our experiment setup.

Algorithm 2 Measurement of services with normally distributed CPU utilisation. Each acquisition took 10 min while the CPU utilisation of each core as well as the overall power consumption were logged.

```
Acquisition of idling system.
for all n ∈ {5, 10, 15, 20, 25, 30} do
    for instance = 1 to 10 do
        Instantiate n randomly chosen services.
        Acquisition of the services managed by the Linux scheduler.
        Acquisition with services randomly assigned to cores.
        Acquisition with services assigned to cores according to Verma et al. [9].
        for all c ∈ {0.5, 0.7, 0.9} do
            Acquisition with services assigned to cores according to meanCSP with effective capacity c.
        end for
        for all ϵ ∈ {0.05, 0.1, 0.25, 0.4} do
            Acquisition with services assigned to cores according to ND-CSP with MEP ϵ.
        end for
    end for
end for
```

We employed the YOKOGAWA WT210 digital power analyser at a rate of approximately 10 Hz in order to measure the overall power consumption of the server. We extracted the average idle power consumption – (187.5 ± 6.2) W – of the system before we carried out our analysis. We employed DSTAT[10] at a rate of 1 Hz in order to log the CPU utilisation of the entire system.

VI. EVALUATION

For each of the 600 scenarios, we extracted the maximum core overload probability from all cores, the average job completion time (JCT), and the average power consumption. Then we averaged each of these three values for all the 10 instances and used them along with their standard deviations for comparison purpose.

[10]http://dag.wiee.rs/home-made/dstat/

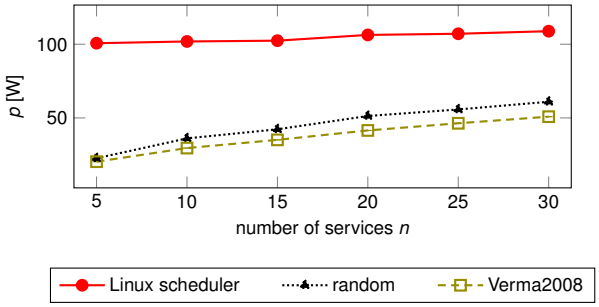


Fig. 6: Comparison of the average power consumption p for the Linux scheduler (all cores active), the random service-to-core assignment (unused cores disabled), and the algorithm proposed by Verma *et al.* [9] (*Verma2008*).

A. Power Consumption

Fig. 6 compares the change in the power consumption of our server as the number of executed services increases. Three scheduling strategies are used for the comparison, namely, the Linux scheduler, the random-to-core assignment, and *Verma2008*. As can be seen, the power consumption of the server was fairly invariable and the largest under the Linux scheduler. Therefore, we shall not refer to it in the subsequent analysis.

In Fig. 7 the power consumption of the server is plotted against the effective capacity c (for meanCSP) and ϵ (for ND-CSP). Since *Verma2008* and *random* are independent of these parameters, the power consumptions for these two schedulers are drawn as horizontal lines. On average, *Verma2008* performed better than ours, but the difference is not appreciable. The *random* scheduler, on the other hand, resulted in a relatively larger power consumption in all the configurations. In this figure, the parameters c and ϵ are not yet correlated. Thus the meanCSP and the ND-CSP strategy cannot be directly compared with one another in terms of their performance. Nevertheless, it can be observed that for some parameter values their power consumptions are lower than the power consumptions of the other two scheduling algorithms. In other words, the CSP as a service consolidation strategy can indeed be applied in order to reduce the power consumption of the server.

An interesting aspect to investigate is the trade-off between the power consumption and the performance of the server during service consolidation. One of the performance metrics from the service provider point of view is the JCT which is a direct consequence of the core overload probability. In order to determine how much of the total capacity we should reserve in order to accommodate the fluctuation of workloads (in other words, in order to minimise the exceeding probability), it is possible to correlate the MEP ϵ in ND-CSP with the effective capacity c in meanCSP. This can be achieved by aligning these parameters to a common criterion, namely, the same power consumption. Therefore, we approximated the dependency of the power consumption p of both CSP strategies on their parameters by a linear regression.

$$p_{\text{meanCSP}}(c) = a_{\text{meanCSP}} \cdot c + b_{\text{meanCSP}} \tag{9}$$

$$p_{\text{ND-CSP}}(\epsilon) = a_{\text{ND-CSP}} \cdot \epsilon + b_{\text{ND-CSP}} \tag{10}$$

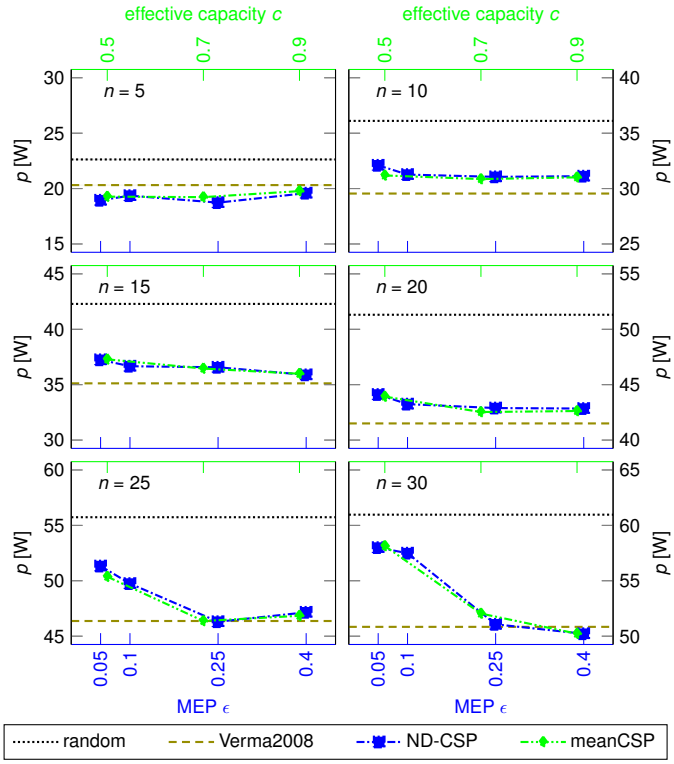


Fig. 7: Comparison of the average power consumption p. Blue squares of the ND-CSP belong to the lower x axis of the maximum exceeding probability (MEP) ϵ and the green diamonds of the meanCSP belong to the upper x axis of the effective capacity c. Both ϵ and c are uncorrelated.

By seeking equality of the two expressions, we get $\tilde{\epsilon}$ which corresponds to the effective capacity c.

$$p_{\text{meanCSP}}(c) \stackrel{!}{=} p_{\text{ND-CSP}}(\tilde{\epsilon}(c)) \quad \Longleftrightarrow \tag{11}$$

$$\tilde{\epsilon}(c) = \frac{a_{\text{meanCSP}} \cdot c + b_{\text{meanCSP}} - b_{\text{ND-CSP}}}{a_{\text{ND-CSP}}} \tag{12}$$

As a result, we can compare the maximum core overload probability and the JCT of the two strategies when consuming the same amount of power. This approach has an additional significance: When executable services have static workloads, the meanCSP yields a near optimal consolidation solution in terms of reducing the power consumption of the server. By identifying the point where ND-CSP achieves a comparable reduction of the power consumption, it is possible to define ϵ, so that ND-CSP can be used to schedule services with stochastic workloads.

Table I and II summarise the normalised power consumption of the meanCSP and ND-CSP. Compared to the Linux scheduler, these two schedulers reduced the power consumption of the server by more than 61 %. By contrast, they increased the power consumption by about 5 % compared to *Verma2008*. But this increase is negligible considering the relatively high penalty the latter introduced on the JCT.

B. Maximum Core Overload Probability

Fig. 8 displays the relationship between the maximum core overload probability (c.o.p.) and the MEP for different numbers of consolidated services under the condition that the

Table I: The relative power consumption of the meanCSP strategy averaged for all numbers of jobs n.

reference	$c = 0.5$	$c = 0.7$	$c = 0.9$
Linux scheduler	38.0%	36.0%	35.9%
random	88.5%	84.7%	84.6%
Verma2008	105.9%	101.3%	101.2%

Table II: The relative power consumption of the ND-CSP strategy averaged for all numbers of jobs n.

reference	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.4$
Linux scheduler	38.2%	37.6%	35.9%	35.9%
random	89.0%	87.8%	84.3%	84.7%
Verma2008	106.5%	105.0%	100.8%	101.3%

meanCSP and ND-CSP consumed equal amount of power when these parameters were used during service consolidation. The horizontal lines display the core overload probability for the other schedulers as these are not dependent of c and ϵ. The shaded regions indicate the standard deviations. Since the Linux scheduler aims for achieving the best performance instead of energy-efficiency, it yielded the least maximum core overload probability of 2 %. By contrast, the *random* scheduler yielded the largest core overload probability.

Furthermore, from Fig. 8, it was possible to establish a correlation between the theoretical ϵ and the practical core overload probability, even though exact equating could not be made, since only the *maximum* core overload probability is shown.

C. Job Completion Time

Fig. 9 can be taken as a justification for service consolidation. In the underlying experiment, we varied the number of simultaneously executing services from 5 to 30, at 30, the server already reaching its maximum capacity. Even though there was a considerable difference in the JCT of the different scheduling strategies, the JCT of each strategy, however, remained, by and large, invariable as we varied the number of executed services. An exception to this is *Verma2008*, which resulted in the worst JCT.

Fig. 10 displays how the JCT of each scheduling strategy changed as we varied the configuration parameters for different amount of executed services. Understandably, apart from the CSP strategies, all the rest were not responsive to the change in c and ϵ. The Linux scheduler and the *random* scheduler were always competitive in their performance, since both of them did not set resource utilisation as their prior goal and achieved relatively short JCT. However, the figure demonstrates that by choosing the appropriate effective capacity and the MEP, it is possible to achieve a competitive or even shorter JCT with both CSP strategies. By contrast, *Verma2008* performed the worst in all the configurations for want of taking into account the time-varying nature of incoming workloads.

Tables III and IV summarise the normalised JCT of the CSP schedulers. Compared to *Verma2008*, both schedulers improved the JCT by up to 58 % whereas for $\epsilon \leq 0.1$ our scheduler was competitive with the *random* scheduler and better than the Linux scheduler clearly demonstrating

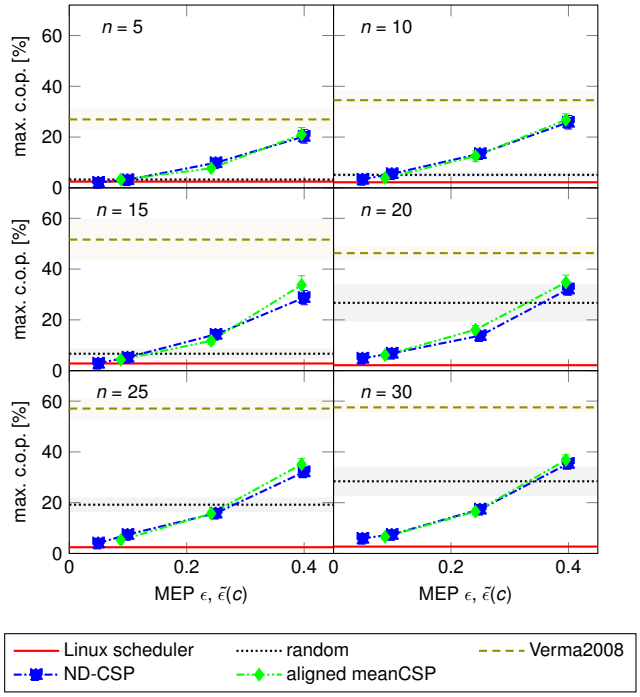


Fig. 8: Comparison of the maximum core overload probability (c.o.p.) for different number of jobs, n. The meanCSP and the CSP strategies are aligned to a common power consumption.

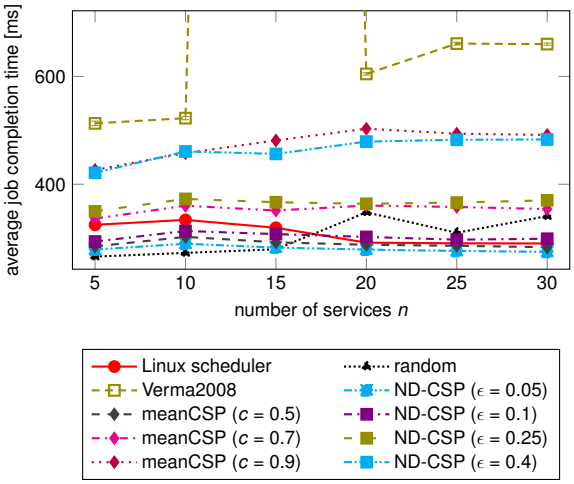


Fig. 9: The job completion time of services with normally distributed workload. The job completion time does not monotonously increase whilst more services are executed between $n = 10$ to 25.

that over-provisioning of resources does not necessarily yield performance improvement.

D. Exponential Workload

As described in Section III, the ND-CSP approximates the resource utilisation statistics of the consolidated services by a normal distribution. In Fig. 11 it can be seen how this approach performs when the services have different resource utilisation statistics (for our case, they had exponentially distributed resource consumptions). From the figure it can be construed that the observation we made with the previous figures for the normal distribution can also be repeated here,

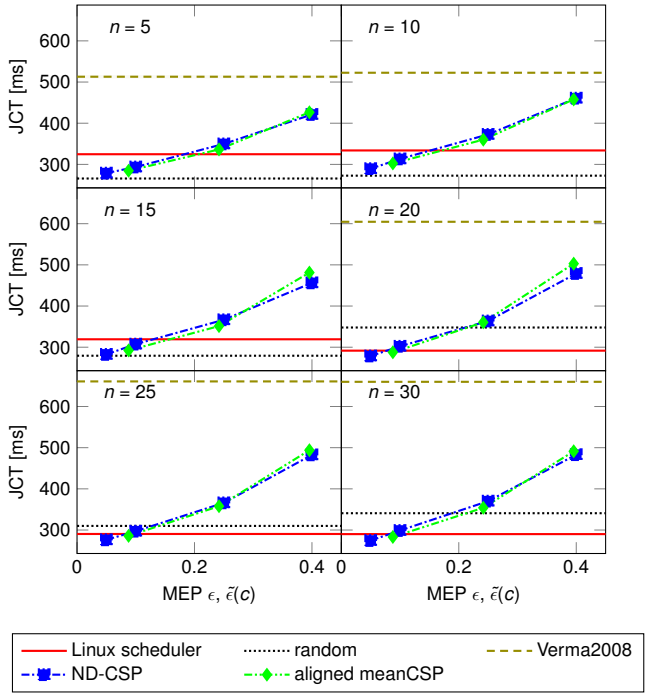


Fig. 10: Comparison of the job completion time (JCT) for the different numbers of jobs n. The meanCSP and the ND-CSP strategies are aligned to a common power consumption.

Table III: Relative JCT of the meanCSP strategy averaged for all numbers of jobs n.

reference	$c = 0.5$	$c = 0.7$	$c = 0.9$
Linux scheduler	94.1%	115.1%	155.2%
random	96.8%	117.9%	158.1%
Verma2008	41.9%	51.2%	68.4%

almost value by value. Consequently, the ND-CSP strategy, which approximates every workload characteristic by a normal distribution, can be used for services having non-Gaussian resource utilisation statistics as well. The implication of this observation is that taking only the means and variances of consolidated services into account is sufficient for employing ND-CSP (i.e., it is not necessary to perform convolution in order to determine the overall resource utilisation density).

VII. Conclusion

In this paper we introduced an extension of the FFD algorithm to consolidate services with stochastic workloads and experimentally validated the algorithm by employing real-world workload (CPU utilisation) traces which we obtained from a Google data centre hosting more than 32 000 virtual machines. We compared the performance of our algorithm with four existing consolidation and scheduling strategies.

Table IV: Relative JCT of the ND-CSP strategy averaged for all numbers of jobs n.

reference	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.4$
Linux scheduler	91.1%	98.3%	118.8%	151.3%
random	93.7%	101.0%	121.8%	154.3%
Verma2008	40.6%	43.7%	52.8%	67.1%

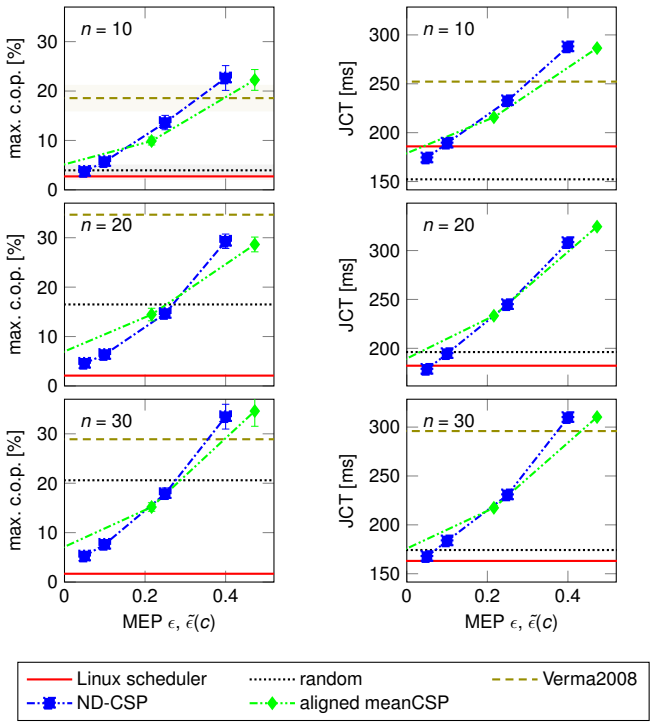


Fig. 11: Comparison of the maximum core overload probability (c.o.p.) and the job completion time (JCT) for the exemplary number of jobs n when services' resource consumptions are exponentially (instead of normally) distributed. The meanCSP and the ND-CSP strategies are aligned to a common power consumption.

Our algorithm takes the overall resource utilisation statistics to compute the maximum exceeding (overload) probability during consolidation. Interestingly, our approach was also useful to identify the effective capacity of a computing node, so that a deterministic FFD can yield a comparative result. In other words, by determining the effective capacity during consolidation, only the expected resource utilisation of the services being computed suffices to carry out a consolidation with FFD without suffering from or introducing an overloading condition.

We employed a 16-core multi-socket system for our experiment. The system served well to consolidate up to 30 services but beyond this number, the effect of overload was conspicuous. On the other hand, we have observed that our consolidation strategy and, in general, the CSP performs very well for a large number of services because it is easier to establish denser patterns with a large number of services having high variability of workloads. Furthermore, the way the cores of the server were supplied with power provided us with little possibility to power down idle cores. Each socket has its own Phase-Locked Loop (PLL), but individual cores in each socket could not be powered down separately or operated with different clock frequencies. As a result, the power efficiency of the consolidation strategy could be demonstrated only marginally. Our recommendation to hardware designers is that every core of a multi-core, multi-socket system should have a separate PLL and should be physically turned down in order to guarantee greater flexibility during dynamic power management.

Our endeavour was invested to optimise the CPU utilisation of a server, as it is the most critical resource for many execution assignments and the most power consuming subsystem. Our approach can be extended to optimise the utilisation of multiple resources at the same time, such as IO and MEM. One way would be to extend the so-called *First Fit Decreasing Height* [32]. The success of this and similar approaches depends on, of course, the extent to which the resources can be regarded, statistically speaking, as independent. Another possibility is to jointly optimise the energy cost of data centre networks, as, indeed, some have already started to do so [33]. In future, we aim to closely examine these and similar aspects.

REFERENCES

[1] S. Cisco Inc., *Global data center IP traffic from 2012 to 2020, by data center type (in exabytes per year), Statista 2018*, 2018.

[2] S. S. Manvi and G. Krishna Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, no. 1, pp. 424–440, 2014.

[3] C. Mobius, W. Dargie, and A. Schill, "Power consumption estimation models for processors, virtual machines, and servers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1600–1614, 2014.

[4] W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, 2015.

[5] F. Zhang *et al.*, "Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 338–351, 2014.

[6] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *Journal of Parallel and Distributed Computing*, vol. 79, pp. 3–15, 2015.

[7] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, 2014.

[8] L. V. Kantorovich, "Mathematical Methods of Organizing and Planning Production," *Leningrad State University*, 1939.

[9] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," *Lecture Notes in Computer Science*, vol. 5346 LNCS, 2008, pp. 243–264.

[10] L. Grigoriu and D. K. Friesen, "Approximation for scheduling on uniform nonsimultaneous parallel machines," *Journal of Scheduling*, pp. 1–8, 2016.

[11] S. G. Kim, H. Eom, and H. Y. Yeom, "Virtual machine consolidation based on interference modeling," *The Journal of Supercomputing*, vol. 66, no. 3, pp. 1489–1506, 2013.

[12] A. Forestiero, C. Mastroianni, M. Meo, G. Papuzzo, and M. Sheikhalishahi, "Hierarchical Approach for Efficient Workload Management in Geo-Distributed Data Centers," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 97–111, 2017.

[13] R. W. Ahmad *et al.*, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[14] X. Yi, F. Liu, D. Niu, H. Jin, and J. Lui, "Cocoa: Dynamic container-based group buying strategies for cloud computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 2, no. 2, p. 8, 2017.

[15] L. Yu *et al.*, "Stochastic Load Balancing for Virtual Resource Management in Datacenters," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[16] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, 2012, pp. 1397–1420. arXiv: 1006.0308.

[17] F. Xu *et al.*, "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Transactions on Computers*, vol. 63, 2014, pp. 3012–3025.

[18] F. Eichhorn, W. Dargie, C. Möbius, and K. Rybina, "HAECubie: A Highly Adaptive and Energy-Efficient Computing Demonstrator," *24th International Conference on Computer Communications and Networks (ICCCN 2015)*, 2015.

[19] S. Homsi *et al.*, "Workload consolidation for cloud data centers with guaranteed QoS using request reneging," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2103–2116, 2017.

[20] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, "Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, 2017.

[21] W. Dargie, "Analysis of the power consumption of a multimedia server under different DVFS policies," *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, IEEE, 2012, pp. 779–785.

[22] M. R. Garey and D. S. Johnson, "'Strong' NP-Completeness Results: Motivation, Examples, and Implications," *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, 1978.

[23] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *European Journal of Operational Research*, vol. 255, no. 1, pp. 1–20, 2016.

[24] J. M.V. V. de Carvalho, "LP models for bin packing and cutting stock problems," *European Journal of Operational Research*, vol. 141, no. 2, pp. 253–273, 2002.

[25] J. Martinovic, G. Scheithauer, and J. M.V. V. de Carvalho, "A Comparative Study of the Arcflow Model and the One-Cut Model for one-dimensional Cutting Stock Problems," *European Journal of Operational Research*, vol. 266, no. 2, pp. 458–471, 2018.

[26] G. Dósa, R. Li, X. Han, and Z. Tuza, "Tight absolute bound for First Fit Decreasing bin-packing: FFD (L) $\leq$ 11/9 OPT (L) + 6/9," *Theoretical Computer Science*, vol. 510, pp. 13–61, 2013.

[27] J. Martinovic, M. Hähnel, G. Scheithauer, W. Dargie, and A. Fischer, "Cutting Stock Problems with Nondeterministic Item Lengths," Preprint MATH-NM-04-2017, Technische Universität Dresden, 2017.

[28] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[29] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," *IEEE Global Telecommunications Conference (GLOBECOM)*, 2012, pp. 2505–2510.

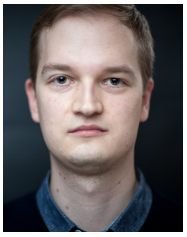
[30] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," *INFOCOM*, 2011, pp. 71–75.

[31] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The mälardalen WCET benchmarks: Past, present and future," *Open Access Series in Informatics (OASIcs)*, vol. 15, no. Wcet, pp. 136–146, 2010.

[32] E. G. Coffman Jr, M. R. Garey, D. S. Johnson, and R. E. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 808–826, 1980.

[33] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," *INFOCOM, 2014 Proceedings IEEE*, IEEE, 2014, pp. 2598–2606.

Markus Hähnel is a doctoral candidate with the Chair of Computer Networks at the Technische Universität Dresden (Germany). He holds MSc degree (2015) and BSc degree (2013) from the Technische Universität Dresden, both in Physics. His research interest is related to energy-efficient computing and data centres. He can be reached at: markus.haehnel1@tu-dresden.de.



John Martinovic received his BSc degree and his MSc degree in mathematics both from Technische Universität Dresden (Germany), in 2012 and 2014. He is currently working as a doctoral candidate at the same university within the fields of discrete optimisation and operations research. In particular, his research focusses on cutting and packing problems and their applications to wireless communications and energy-efficient computing. He can be reached at: john.martinovic@tu-dresden.de.

Guntram Scheithauer obtained his PhD in mathematics from Technische Universität Dresden (Germany) in 1983, and has been working as a research associate at the Institute of Numerical Mathematics since the same year. His research interest is related to discrete and combinatorial optimisation, particularly to the theory and applications of cutting and packing problems. He can be reached at: guntram.scheithauer@tu-dresden.de.

Andreas Fischer is professor of numerical optimisation with the Technische Universität Dresden, Germany. Since 2013 he is the director of the Institute of Numerical Mathematics. His research concentrates on the design and analysis of efficient algorithms within the field of mathematical programming, particularly complementarity systems, generalised Nash equilibrium problems, discrete optimisation with applications, and machine learning. He can be reached at: andreas.fischer@tu-dresden.de.

Alexander Schill is a professor for Computer Networks at Technische Universität Dresden, Germany. His major research interests include distributed system architectures, advanced web technologies, and mobile and ubiquitous computing. Prof. Schill holds M.Sc. and Ph.D. degrees in Computer Science from the University of Karlsruhe, Germany, and received a Doctor Honoris Causa from Universidad Nacional de Asuncion, Paraguay. Before his tenure in Dresden, he also worked at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. He can be reached at: alexander.schill@tu-dresden.de.

Waltenegus Dargie (SM'12) is an Associate Professor and leads the Energy Lab at the Technische Universität Dresden. He obtained a PhD in Computer Engineering from the Technische Universität Dresden in 2006 and holds MSc degree (2002) from the Technical University of Kaiserslautern (Germany) and BSc degree (1997) from the Nazareth Technical College (Ethiopia), both in Electrical Engineering. Dr. Dargie is a senior member of the IEEE and a member of the editorial board of the Journal of Computer Communication. His research interest is related to stochastic processes, energy-efficient computing, and ubiquitous computing. He can be reached at: waltenegus.dargie@tu-dresden.de.