Energy-Aware Workload Management Models for Operation Cost Reduction in Data Centers

Abstract

In the last century, the costs of powering datacenters have increased so quickly, that datacenter power bills now dwarf the IT hardware bills. Many large infrastructure programs have been developed in the past few years to reduce the energy consumption of datacenters, especially with respect to cooling requirements. Although these methods are effective in lowering the operation costs they do require large upfront investments. It is therefore not surprising that some datacenters have been unable to utilize the above means and as a result are still struggling with high energy bills. In this work we present a cheap addition to or an alternative to such investments as we propose the use of intelligent, energy efficient, system allocation mechanisms in place of current packaged system schedulers available in modern hardware infrastructure cutting server power costs by 40%. We pursue both the quest for (1) understanding the energy costs generated in operation as well has how to utilize this information to (2) allocate computing tasks efficiently in a cost minimizing optimization approach. We were able to underline the energy savings potential of our models compared to current state-of-the-art schedulers. However, since this allocation problem is complex (NP-hard) we investigated various model approximations in a trade-off between computational complexity and allocative efficiency. As a part of this investigation, we evaluate how changes in system configurations impact the goodness of our results in a full factorial parametric evaluation.

Keywords: *OR in Service Industries; OR in Telecommunications; Decision Support System; Simulation; Cost Management;*

1 Introduction

Competition in the computation and storage industry is fierce, with names like Amazon and Oracle setting radically low pricing levels for customers, ultimately dominating the cloud computing industry. If classic datacenters are to survive in the increasing trend towards cloud computing, they need to match these prices. With power bills dwarfing the IT hardware bills, organizations continuously search for management solutions to reduce power consumption in their datacenters[1].

In terms of 'effort-to-implement', currently discussed solutions can be categorized into long-term strategic, intermediate managerial and short-term operational activities. Long-term activities are associated with the location of the datacenters. Relocation strategies require long term investments and lead time of at least six months for smaller datacenters, which makes them unattractive as a tool in the short run. A vivid example is given by Google, who built their new datacenter near the hydroelectric facility in Dalles, Oregon. Here the operation costs could be reduced due to the availability of inexpensive hydroelectric power. Other companies explored the possibility to relocate their datacenters to Alaska, in an attempt to reduce the need for cooling. These extreme temperatures, however, also may require heating the datacenter facilities when temperatures fall too low. Most of the companies have dropped the idea of hosting datacenters in arctic regions also for reasons of *connectability*. The above solutions are examples of decisions which have far-reaching consequences and are thus not an instrument for active cost reduction required in the near future.

Intermediate managerial measures typically address the architectural design of datacenters. The most prominent example of intermediate measures is the installation of hot-cold aisle configurations of server and cooling vent placement. Other intermediate means could be replacing current hardware with more costly equipment with higher efficiency. An example would be to switch from an air-based cooling system to a liquid cooling system. Modern hardware such as servers that utilize liquid cooling are readily available but again, very costly. Regardless, such changes in the architectural design of datacenters can be implemented in a short time and if operators are willing to spend a large amount investing in such solutions, they are effective as they may attain a minimum of green diligence.

Although long-term and intermediate efficiency measures are effective in lowering the operation costs they do require large upfront investments. It is therefore not surprising that

[1] IT Management: http://www.itmanagement.com/features/datacenter-power-consumption-061907/

some datacenters have been unable to utilize the above means and as a result are still struggling with high energy bills.

One inexpensive short-term method to cut down energy costs is removing the inefficiencies present in operation of computing devices in terms of allocation. By implementing power-aware allocation of applications or jobs, we can simultaneously address the underutilization of servers and further decrease the overall energy costs by migrating jobs to better use the servers and switch the freed-up servers to a lower power state (Dasgupta, A. Sharma, Verma, Neogi, & Kothari, 2011). This solution has the advantage of being inexpensive and easy to implement, as 'merely' the VM scheduler needs to be reconfigured, allowing both small and large corporations to reap the benefits. For this reason in this work we will pursue efficiency by designing an energy efficient scheduler to regulate which computing task should be allocated to which system in attempt to reduce the costs of operation of every server and the datacenter as a whole.

Generally, allocation problems, specifically those that cover the placement of tasks on resources, can be divided into two distinct groups of problems, namely the case of *constrained resources* and *abundant resources*. The quest behind *resource constrained allocation* is the decision which task to accept and allocate to the constrained amount of resources, and according to (Bottcher, Drexl, Kolisch, & Salewski, 1999) has received considerable attention from researchers in the past. The pursuit in *resource abundant allocation* problems however is not the prioritization of tasks, but rather a decision which of the ample resources should be used - herein lays the potential for energy efficiency. In the constrained problem we are only able to maximize our rewards given a fixed amount of energy, but in the abundant resources scenario we can minimize the used resources given a task package. We cover the case of *resource abundant allocation*, which according to related work is often the case for modern datacenters.

First approaches were discussed by (Weiser, Welch, A. Demers, & Shenker, 1994) in form of exponential power models based on battery-operated systems, addressing energy consumption on a chip-level and following (Yao, A. Demers, & Shenker, 1995) the theoretical study of speed scaling policies to manage energy was initiated. It concluded that less energy is consumed over the duration of a process if the speed of the processors is reduced, as confirmed in more recent work by (Albers, 2010).

This insight was restricted to a chip-level analysis. (See 2008) found that most energy flowing into systems is given off as heat and as a result only about 10% (percentage varies in hardware complexity) of energy is actually available for computing. Expanding the analysis to an entire system therefore suggests a decreasing function power model. This is in accordance to (Pisharath, Choudhary, & Kandemir, 2004; Wang & M. Chen, 2008), who in consolidation of server resources suggest power management for single server environments, combining energy-efficiency and management of servers minimizing the energy consumed by a system, while meeting an execution time constraint. This design is also present in recent work by (Park & Pu, 2007; Pinheiro, Bianchini, Carrera, & Heath, 2003; Son, Malkowski, G. Chen, Kandemir, & P. Raghavan, 2007) for server cluster environments. Expanding the analysis to a package of interdependent tasks, (G. Chen, Malkowski, Kandemir, & S. Raghavan, 2005) pursue energy efficiency in process chains, by reducing the voltages/frequencies of processors executing tasks that are not in the critical path of the process.

(J. S. Chase, Anderson, Thakar, Vahdat, & Doyle, 2001; Freeh et al., 2007; Hamann, 2008; Nathuji, Isci, Gorbatov, & Schwan, 2008; Raghavendra, 2008; Rivoire, Shah, Partha Ranganathan, & Kozyrakis, 2007) proposed solutions for server clusters and datacenters, ranging from energy-saving power supplies, to the powering down of unused hard drives to using different power states to conserve energy. Other work has focused mainly on turning idle machines off, to save power as shown in (Burge, Partha Ranganathan, & Wiener, 2007; Mastroleon, Bambos, Kozyrakis, & Economou, 2005) for datacenters. More specifically, (Burge, Partha Ranganathan, & Wiener, 2007) showed that it matters which machines are assigned to each customer, especially when the data center is under saturated and that simple heuristics like turn off a machine as soon as it becomes idle, can save a lot of money. Recent observations by hardware vendors however show that turning servers off, could result in the systems not coming back online and disadvise turning server machines off but to rather send idle resources to sleep, a feature standardized in modern computing equipment. This has the added benefit of being able to power the machines quicker, and reduces failure rates.

Collecting the above insights gained from related literature, we propose the use of intelligent, energy efficient, system allocation mechanisms in place of the current packaged system schedulers available in modern hardware infrastructure as proposed by (Dasgupta, A. Sharma, Verma, Neogi, & Kothari, 2011). Complementing their findings, we explore the energy savings potential in an in-depth analysis, by simulating various different infrastructure configurations in an attempt to find exactly by how much we can reduce our energy consumption when implementing energy efficient resource schedulers.

This work is divided as follows. In section two we discuss and motivate the research methodology followed by this work. In section three we present the model and the various model heuristics to be used to solve the resource cost minimization problem. Section four contains the numerical evaluations. Here we analyze the complexity, solution quality of each of our models, as well as show some insights gained from the simulations before we conclude with management-centric insights, recommendations and outlook in section five.

2 Methodology

"How effective are energy minimizing resource allocation algorithms compared to current allocation mechanisms and at what cost (in terms of runtime complexity) can they be implemented?" This is the primary research question behind this work. We construct a model benchmark calculating the best possible allocation to minimize the total energy consumed by a datacenter, subject to a number of necessary operational constraints, as a mathematical optimization problem. To construct such a model, we must first understand how allocating a task to a server affects its energy consumption. We therefore formulate our secondary research question required to fully address the primary question: *"How is energy consumed by datacenter servers and how can we map them?"*

2.1 Describing Server Energy Consumption

Before addressing the central research question of allocation, we must first explore the secondary research question of system energy consumption. To do this we used the published results from the Standard Performance Evaluation Corporation (SPEC) to formulate and evaluate the energy consumption of servers. Figure 1 graphically displays the normalized power consumption relationship, with the highest and lowest recorded values are depicted by the bars and the crosses mark the average over 183 by recorded server types featuring benchmark trials on systems (2008 – 2011)). The vertical axis depicts the power costs normalized between 0 and 1. The horizontal axis depicts the recorded categories of utilization as recorded by SPEC. The figure shows a power/utilization ratio in idle mode to range between 25% and 75% for the tested systems. This is in accordance to (Moore, J. Chase, P Ranganathan, & R. K. Sharma, 2005) who found that idle machines consume roughly 50% of the power compared to those at full utilization. Further, we found traces of a non-linear trend.

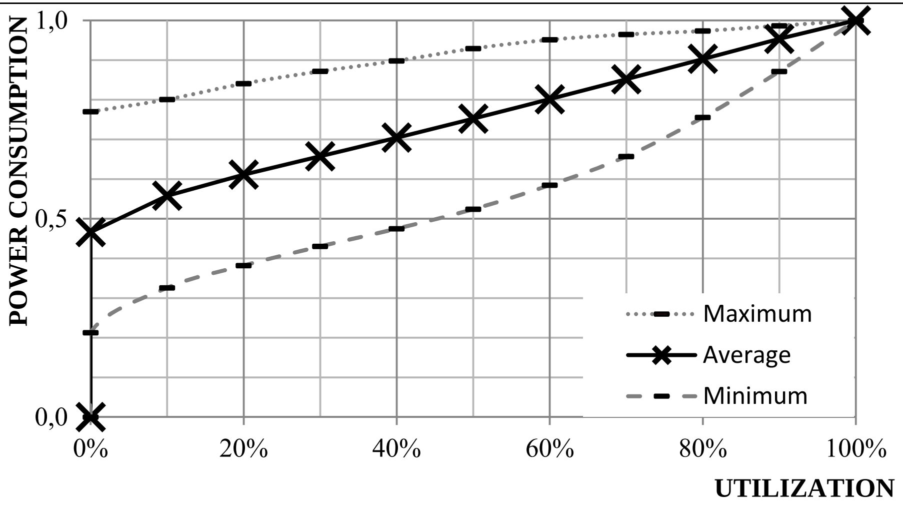


Figure 1. Relative power consumption of servers graphed against utilization

Based on the observations in Figure 1, we form the following proposition assuming:

Proposition 1: *The dependency of utilization and power consumption is best captured by the non-linear allometric power function of the form $C = a X^{b} + d$.*

To validate the above proposition, we performed a series of regression analysis on the available SPEC data for each of the 183 systems. Table 2 presents an abridged summary of the coefficients of determination (the measure of how well the regression line approximates the real data points) for linear, exponential, logarithmic and allometric regressions:

Table 1. R2 Results Table for Functional Analysis

	Logarithmic	Exponential	Linear	Allometric
	$C = (b \ln(X)) + d$	$C = a e^{(b X)} + d$	$C = a X + d$	$C = a X^{b} + d$
R^2 Minimum	0,2218	0,7996	0,8599	0,9727
R^2 Maximum	0,6542	0,9994	0,9997	0,9999
R^2 Average	0,3686	0,9432	0,9856	0,9935
R^2 Median	0,3721	0,9655	0,9877	0,9959

According to the above table (Table 1), comparing the regressive fit of each function to the observed data points the empirical analysis supports our proposition that an allometric function best describes the dependency as the R^2 values are the highest, closely followed by the linear function. To ensure that this higher R^2 result is not due to over fitting, we additionally compare the functions using the modified Akaike information criterion AIC_{R^2}. The modified AIC_{R^2} is a measure of the goodness of fit of a statistical model. Accounting for parsimony, it allows selecting the best function to describe the observed relation without the bias of model over fitting. For a given set of observations and resulting functions, the preferred function is the one with the lowest $\mathrm{AIC}_{\mathrm{R}^2}$ value. Formally, $AIC_{R^2} = n \, ln \frac{1-R^2}{n} + 2k$, where k is the number of parameters in the statistical model and n are the number of observations.

Table 2. $\mathrm{AIC}_{\mathrm{R}^2}$ Values for Functional Analysis

	Logarithmic	Exponential	Linear	Allometric
	C = (b ln(X)) + d	C = a $e^{(b\,X)}$ + d	C = a X + d	C = a X^{b} + d
AIC_{R^2}[2]	-21,533	-33,10	-38,679	-53,034

Table 2 shows the $\mathrm{AIC}_{\mathrm{R}^2}$ values for each model function type. Since the function with the lowest AIC is the preferred one, the allometric function is a better fit than the linear function estimate. As a result, we can confirm the correctness of proposition 1, and the allometric function captures the dependency of utilization and energy consumption best.

2.2 Model Development

As shown above the allometric function best captures the dependency of energy costs on utilization. We denote the model using the allometric function as NLINFIX, abbreviating the fact that we have a non-linear (i.e. allometric) optimization problem with a fixed cost component. Unfortunately, since NLINFIX is a non-linear binary problem, it is to be expected that solving the problem will be computationally intensive. As a result, we chose to perform a series of model approximations which trade computational complexity at the expense of increased costs. Our model heuristics are devoted to solve the master optimization problem by approximating the cost functions defining sub-optimization problems that are easier to solve. These model approximations are summarized as cost-oriented model approximations in section 2.2.1. To properly evaluate our approximations impact, we compare our model approximation solutions to those found by technical, utilization-oriented, allocation heuristics that are currently used in practice. These allocation heuristics are summarized in a single utilization-based model heuristic in section 2.2.2.

[2]

2.2.1 *Cost-oriented model approximations*

From the structure of our allometric cost function we can identify two main factors affecting the complexity. The first and most obvious complexity lies in the concave non-linearity of the optimization problem. The first model heuristic we will use is the relaxation of this non-linearity. As we have seen from our previous empirical analysis, we know that the approximation quality is around 10 %. Thus, we expect our linear heuristic to be close to the allometric model.

The second issue refers to the binary nature of the optimization problem, which controls the *on/off decision* during the optimization process. Consider the following example which explains this intuition: Assume a task is to be executed on one of two nodes. Node A costs €10 to run while node B costs €1 to operate. These are the utilization independent costs and occur if a node is powered regardless of whether an application is executed on the node or not. Executing the task on a node causes further costs, namely €1 if the task is executed on node A and €5 if it is executed on node B. These are the so-called utilization dependent costs. The choice is now, which of the two is cheaper and thus which should be selected using an optimization approach. An optimization model without the discussed on/off decision would look only at the utilization dependant costs and select node A. Node B would therefore be shut down *post-optimization*, since it is not needed. The total costs of this model would therefore amount to €11 (€1 variable + €10 fixed). A optimizer including the on/off decision in its optimization process would look at the whole picture, making it a little more complex, but would look beyond only the variable costs and find that node B is cheaper to operate since it only costs €6 (€5 variable + €1 fixed) in total to operate.

Mathematically, including this process in optimization is modeled using

COST = $aUTIL^b + dY$, where $d > 0$ and $Y = \text{sign}(UTIL)$. Therefore,

$$COST = \begin{cases} 0, & UTIL = 0 \\ aUTIL^b + dY, & UTIL > 0 \end{cases}$$

Figure 2 shows the model matrix, in relation to the *non-linearity* property or the *on-off decision* capability.

Non-linearity	Yes	NLINFIX $(a\,UTIL^b + d)$	NLIN $(a\,UTIL^b)$
	No	LINFIX $(a\,UTIL + d)$	LIN $(a\,UTIL)$
		Yes On/Off	No

Figure 2. Model Complexity Breakdown

NLINFIX in the upper left corner features both properties. The second model we introduce, NLIN features the first in a series of model heuristics, as we remove the on/off decision from optimization. The third model LINFIX relaxes the non-linearity, allowing for much faster calculation as a linear integer problem by approximating the saleable, utilization dependant power costs as a linear function. The fourth model LIN (section 3.5) combines the approximations used in NLIN and LINFIX, excluding the on/off decision and relaxing the non-linearity. These four models are referred to as the cost-oriented model heuristics.

2.2.2 Utilization-oriented benchmark heuristic

When introducing new concepts for any process already practiced by the industry, it is important to always set a benchmark to compare against. Current system schedulers are often based on the optimization of the utilization of single systems, or the overall utilization of all systems under the schedulers' control, which is the case for current system schedulers, such as LSF[3], LoadLeveler[4], NQS[5] or TORQUE[6]. (Vengerov, 2009) implemented these principles in his work, labeling this practice as 'BESTFIT' allocation which places the applications on nodes in such a way, as to maximize the average utilization over all nodes. In our case we implement the dual version of the problem and minimize the leftover utilization potential (i.e. the node capacity left unused) according to the 'no waste' principle and label our implementation of his algorithm as BESTFIT (see section 3.6 for a full model description). Table 3 shows a summary of the Model Attributes Overview described in section 2.2.

Table 3. Model Attributes Overview

Model Name	Cost-oriented			Utilization
	Non-Linear Cost Function	Linear Cost Function	On/Off Decision	
NLINFIX	√		√	
NLIN	√			
LINFIX		√	√	
LIN		√		
BESTFIT				√

[3] http://www.platform.com/workload-management/high-performance-computing

[4] http://www-03.ibm.com/systems/software/loadleveler/

[5] http://gnqs.sourceforge.net/docs/starter_pack/introducing/index.html

[6] http://www.clusterresources.com/products/torque-resource-manager.php

3 The Models

In the next section, we proceed to describe each of the introduced models formally, as well as describe their functionality with a short example study. We begin by introducing and explaining each of the variables used in the following models.

3.1 Parameter Descriptions

Common to all models are a set of parameters, which are defined as follows:

Set of all computing nodes	n
Computing units supplied by node n	cs_n
Memory units supplied by node n	ms_n
Exogenous cost function parameters of node n	a_n, b_n, d_n
Cost component: Variable, utilization dependant	$kvar_n$
fixed, if node active	$kfix_n$
Set of all computing tasks to be executed	j
Computing units required by j	cd_j
Memory units required by j	md_j
First timeslot where application j is to be executed	$first_j$
Last timeslot where application j is to be executed	$last_j$
Decision variable if application j is to be executed on node n in time t	x_{jnt}
Decision variable if node n is to be powered time t	y_{nt}

To simplify the understanding of the data parameters and their functionality, we have generated a data sample as shown in Table 4.

Table 4. Data Sample

App	cd_j	md_j	$first_j$	$last_j$	Node	cs_n	ms_n	a_n	b_n	d_n	k_{fix}	k_{var}
J1	30	45	1	4	N1	150	249	3,90	0,47	3,00	3,89	3,35
J2	25	31	2	3	N2	145	259	4,50	0,28	3,00	4,67	3,34
…					…							

The table can be read as follows: J1 is a request for a, application to be run from timeslot 1 to 4 and requires a minimum of 30 CPU's and 45 units of memory in each timeslot. Node N1 offers 150 CPU's and 249 units of memory and costs approximately 3.89 monetary units (MU) per 10% increase in utilization and timeslot. $kvar_n$ is the linear approximate of the cost function and is required for the linear models. $kfix_n$ is the y-intercept to the linear estimate. $kvar_n$ and $kfix_n$ are derived from the non-linear function parameters a_n, b_n and d_n. The respective solution of each model is then plugged into the equation (CF) which resembles the total costs ‘as if it were implemented and measured’.

$$K_{Model} = \sum_n^N \sum_t^T \left(a_n * \left(\frac{\sum_j^J x_{jnt} cd_j}{cs_n} \right)^{b_n} + d_n * sign\left(\sum_j^J x_{jnt}\right) \right) \qquad \text{(CF)}$$

This allows the resulting allocation of each model to be comparable. In the next sections 3.2-3.6 we formally describe the various models introduced in this work.

3.2 Base Model: NLINFIX

NLINFIX is defined as follows:

$$min_x\, K = \sum_n^N \sum_t^T \left(a_n * \left(\frac{\sum_j^J x_{jnt} cd_j}{cs_n} \right)^{b_n} + d_n * y_{nt} \right) \qquad \text{(NF1)}$$

Subject to:

$$x_{jnt} \in \{0,1\}, \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T \qquad \text{(NF2)}$$

$$\sum_n^N x_{jnt} = \begin{cases} 1, & \forall\, j \in J, \forall\, t \in [first_j; last_j] \\ 0, & \forall\, j \in J, \forall\, t \in T \backslash [first_j; last_j] \end{cases} \qquad \text{(NF3)}$$

$$\sum_j^J x_{jnt} * cd_j \;\le\; cs_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(NF4)}$$

$$\sum_j^J x_{jnt} * md_j \;\le\; ms_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(NF5)}$$

$$y_{nt} \in \{0,1\},\;\; sign\left(\sum_j^J x_{jnt}\right) = y_{nt} \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(NF6)}$$

In NLINFIX, allocation is determined by the binary decision variable x_{jnt}, where $x_{jnt} = 1$ if application j is allocated to node n in time slot t, and $x_{jnt} = 0$ if not. Equation NF1 shows the allometric objective function of NLINFIX which minimizes costs based on the utilization of CPU’s and fixed costs incurred if nodes are active. The cost function is dependent on the current utilization and for each active node, d_n are added for each node y_{nt} used. By design, those nodes left unused are not considered since they can, and by assumption are, automatically shut down at no further cost. We have opted to use CPU’s only in our target function, since most costs are generated through the operation of cores. For simplicity we assume that these costs indirectly include all costs of operation (i.e. cooling, network costs etc.). Equation NF2 introduces the binary decision variable x and NF3 ensures the enforcement of agreements allocating all applications in the schedule and ensures each application is only executed on one node at a time and t is only defined for those time periods where allocation for the given combination of applications and nodes is actually feasible. Equations NF4 and NF5 are the resource constraints and ensure that allocations do not exceed the capacity of the systems. Finally, equation NF6 binds y_{nt} forcing the fixed costs to be

included only when a application is allocated to a node. Should no application be allocated to a node, no costs are generated as the node is not powered in that time slot.
In the following sections we will refer to the NLINFIX allocation as a benchmark solution, comparing the allocation with the model heuristics and pointing towards potential flaws in allocation of each approximation.

3.3 Model Approximation: NLIN

Let NLIN be defined as follows:

$$min_x\, K = \sum_n^N \sum_t^T \left(a_n * \left(\frac{\sum_j^J x_{jnt} cd_j}{cs_n} \right)^{b_n} \right) \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T \qquad \text{(N1)}$$

Subject to:

$$x_{jnt} \in \{0,1\}, \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T \qquad \text{(N2)}$$

$$\sum_n^N x_{jnt} = \begin{cases} 1, & \forall\, j \in J, \forall\, t \in [first_j; last_j] \\ 0, & \forall\, j \in J, \forall\, t \in T \backslash [first_j; last_j] \end{cases} \qquad \text{(N3)}$$

$$\sum_j^J x_{jnt} * cd_j \ \le\ cs_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(N4)}$$

$$\sum_j^J x_{jnt} * md_j \ \le\ ms_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(N5)}$$

In NLIN we introduce the first of a series of approximations described above. Here we relax the requirement for the model to include the fixed costs of each node in the objective function N1, hence reducing the model complexity by the second term in the target function, as well as removing the last equation NF6 since constraining y_{nt} is no longer necessary. The remaining constraints N2 – N5 are kept the same as the benchmark equations NF2 – NF5.

3.4 Model Approximation: LINFIX

Let LINFIX be defined as follows:

$$min_x\, K = \sum_n^N \sum_t^T \left(kvar_n * \left(\frac{\sum_j^J x_{jnt} cd_j}{cs_n} \right) + kfix_n * y_{nt} \right) \qquad \text{(LF1)}$$

Subject to:

$$x_{jnt} \in \{0,1\}, \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T \qquad \text{(LF2)}$$

$$\sum_n^N x_{jnt} = \begin{cases} 1, & \forall\, j \in J, \forall\, t \in [first_j; last_j] \\ 0, & \forall\, j \in J, \forall\, t \in T \backslash [first_j; last_j] \end{cases} \qquad \text{(LF3)}$$

$$\sum_j^J x_{jnt} * cd_j \ \le\ cs_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(LF4)}$$

$$\sum_j^J x_{jnt} * md_j \ \le\ ms_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(LF5)}$$

$$y_{nt} \in \{0,1\},\ \ sign\left(\sum_j^J x_{jnt}\right) = y_{nt} \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(LF6)}$$

In LINFIX we introduce the second of a series of approximations. Instead of reducing the model complexity by the second decision variable, we relax the nonlinearity of the model by approximating the nonlinear cost function with a linear constant *kvar$_n$* using a simple linear regression of the underlying cost function of the node. (We assume the function of costs is unknown, the utilization is used as a predictor variable and the costs serve as a response variable. Using the ordinary least squares method the marginal effect of utilization on operation costs *kvar$_n$* is determined. *kvar$_n$ can thus be interpreted as the approximate expected change in costs for an increase in utilization.)* Therefore, the objective function LF1, is now a linear function, which aims to minimize costs based on the utilization of CPU's as well as the amount of active nodes. The remaining constraints LF2 – LF6 are kept the same as the benchmark equations NF2 – NF6.

3.5 Model Approximation: LIN

Let LIN be defined as follows:

$$min_x\, K = \sum_n^N \sum_t^T \left(kvar_n * \left(\frac{\sum_j^J x_{jnt} cd_j}{cs_n} \right) \right) \qquad \text{(L1)}$$

Subject to:

$$x_{jnt} \in \{0,1\}, \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T \qquad \text{(L2)}$$

$$\sum_n^N x_{jnt} = \begin{cases} 1, & \forall\, j \in J, \forall\, t \in [first_j; last_j] \\ 0, & \forall\, j \in J, \forall\, t \in \mathrm{T} \setminus [first_j; last_j] \end{cases} \qquad \text{(L3)}$$

$$\sum_j^J x_{jnt} * cd_j \le cs_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(L4)}$$

$$\sum_j^J x_{jnt} * md_j \le ms_n, \qquad \forall\, n \in N, \forall\, t \in T \qquad \text{(L5)}$$

In LIN we combine both approximations. We now reduce the model complexity by both the second decision variable and nonlinearity. Again the nonlinearity is replaced with the marginal effect parameter *kvar$_n$* and the fixed costs term with *y$_{nt}$* is dropped. The remaining constraints LF2 – LF5 are kept the same as the benchmark equations NF2 – NF5, and NF6 is omitted.

3.6 Model Approximation: BESTFIT

To serve as a practical use-case benchmark, we enquired about allocation methods commonly used in datacenters today and found an algorithm as presented by (Vengerov 2009). Following Vengerov's intuition "applications are allocated where they fit best" (SUN/Oracle). In this work we use the simple version of the model and refer to it as BESTFIT (described as such by the authors).

$$min_x\, K = \sum_n^N \sum_t^T \left(\frac{cs_n - \sum_j^J x_{jnt} cd_j}{cs_n} \right) \qquad \text{(BF1)}$$

Subject to:

$x_{jnt} \in \{0,1\}, \qquad \forall\, j \in J, \forall\, n \in N, \forall\, t \in T$ (BF2)

$\sum_n^N x_{jnt} = \begin{cases} 1, & \forall\, j \in J, \forall\, t \in [first_j; last_j] \\ 0, & \forall\, j \in J, \forall\, t \in T \setminus [first_j; last_j] \end{cases}$ (BF3)

$\sum_j^J x_{jnt} cd_j \leq cs_n, \qquad \forall\, n \in N, \forall\, t \in T$ (BF4)

$\sum_j^J x_{jnt} md_j \leq ms_n, \qquad \forall\, n \in N, \forall\, t \in T$ (BF5)

To convert the above intuition into an optimization problem, we opted to minimize the amount of utilization left unused. The solver thus searches for the combination of applications which neatly packs each node. Mathematically, this process is shown by Eq. BF1. The remaining constraints BF2 – BF5 are kept the same as the benchmark equations NF2 – NF5, and NF6 is omitted.

4 Model Evaluations

Generally, two possible approaches can be found in literature when working with test instances. First, there are the practical use cases, which have high practical relevance in use, yet they do not follow any systematic structure required for rigorous analysis (Bottcher et al., 1999). As a result, an algorithm that performs well on one specific practical instance is not guaranteed to perform equally well on other instances. Second, there are artificial numeric simulations, generated randomly given predefined specifications. Their strength lies in the fact that fitting them to certain requirements such as given probability distributions poses no problem. They may however reflect situations with little or no resemblance to problem settings of practical interest. Hence, an algorithm performing well on several such artificial instances may or may not perform satisfactorily in practice. In this work, we attempt to work with the best of both worlds using artificially enhanced practical use cases following the simulation and evaluation approaches described by (Bottcher et al., 1999; Caprara, Fischetti, & Toth, 2002). More specifically, the node parameters were derived from energy cost data found by SPEC to generate 180 node types. The remaining required parameters were generated randomly and are described in the following section 4.1.

4.1 Data Generation

The infrastructure parameters were drawn from the SPEC samples described in section 2.1. The application resource parameters were drawn randomly from a positively skewed lognormal distribution function $\log \mathbb{N}_0^+ (\mu, \sigma^2)$, as recommended by (Feitelson 2002). The distribution parameters are shown in Table 5. Each combination generated using these parameters are referred to as an “instance”. Each order pair is referred to as a “case”.

Table 5. Data Generation Parameter Distribution			
Variable	Variable Description	Source	Distribution
cd_j	Computing units required by j	Generated	Lognormal LogN (3; 1.1)
md_j	Memory units required by j	Generated	Lognormal LogN (5; 1.7)
$first_j$	First timeslot where application j is to be executed	Generated	Uniform [1; 5]
$last_j$	Last timeslot where application j is to be executed	Generated	Uniform [firstj; 5]

For every case and instance all models were subject to the same dataset, allowing the solutions to be directly comparable. For each case, ten instances were generated and the average taken. The simulation was run using GAMS, where the MIP problems were solved using CPLEX and the MINLP problems was solved using the SBB solver with CONOPT3 acting as sub-solver for the NLP sub-problems. While all problem instances covered above belong to the class of NP-hard ones, the computational tractability of a specific instance depends on the problem parameters introduced above.

In order to generate and evaluate the test instances, we used a full factorial design approach, independently analyzing the impact of changing one parameter at a time. In a preliminary study, we found the following parameters to be of interest for evaluation:

Allocation Timeframe (AT) [0,5] determines the horizon |T|;

Order Size (OS) [5,50, step 5] determines the amount of applications (in multiples of 5) ordered to be allocated in |T|;

Resource "Constrainedness" (RC) [5,50, step 5] determines the average amount of nodes (in multiples of 5) readily available in each |T|;

Resource Factor (RF) [0,1, step 0,25] reflects the density ratio of application size and node size. For RF 0,25 the applications are on average a quarter the size of the nodes (with pre-set variances), while for RF 1 the applications and nodes share the same distribution and magnitude.

We generated 10 instances for each combination of AT, OS, RC, and RF which gave a total of 10 x 5 x 10 x 10 x 4 benchmark instances of all five models (equals 20000 generated instances).

4.2 Solution Analysis

The models were run on multiple Intel Dual Core (1.67 GHz, 2.5 GB RAM) without limits for the MIP models. The MINLP models were limited to ten thousand integer solutions found using a combination of DFS (depth first search) and branch and bound algorithms (Solver GAMS/SBB using CONOPT as NLP Solver) or one hour of solver time and the best solution value, and the best lower bound found before this limit were recorded. Should the MINLP models not find a solution within these bounds, the MIP solutions were fed to the MINLP models as an initial solution and subsequently again limited to ten thousand integer solutions found or one hour of solver time. The respective solution of each model was then plugged into the equation (CF) which resembles the total costs 'as if it were implemented and measured' to determine the total costs incurred when using the respective allocation, allowing the models to be comparable in their solution:

$$K_{Model} = \sum_{n}^{N}\sum_{t}^{T}\left(a_n * \left(\frac{\sum_j^J x_{jnt}cd_j}{cs_n}\right)^{b_n} + d_n * sign\left(\sum\nolimits_j^J x_{jnt}\right)\right) \qquad \text{(CF)}$$

We report our solution analysis in tables Table 7, Table **8**, Table 9 and Table 10. Each column contains the following information and displayed as shown in the table key (Table 6):

Table 6. Table Key *(Relevant to Table 7, Table 8, Table 9 and Table 10)*

PAR (T6.1)	**Model Description**		**…** / **…**	**SOL N≻L**	***# Feas***
Instance Size	*Solution's Average Energy Costs incurred (in W)* **(T6.2)**	*% Cost Increase to NLINFIX* **(T6.3)** / *(Coefficient of Variation)* **(T6.4)**	…	*Percent of cases Where NLINFIX Solutions ≻ LINFIX Solutions* **(T6.5)**	*Number of feasible generated Instances* **(T6.6)**

The leftmost column **PAR (T6.1)** indicates the observed effect for the respective parameter, as well as the instance characteristic associated with it. For each model and instance, three values are documented as displayed in Table 6, namely: **Average (T6.2)** in the left cell; **% Cost Increase (T6.3)** to NLINFIX in the upper right cell, showing the additional amount of energy required to operate the infrastructure using the respective model instead of NLINFIX; **Coefficient of Variation (T6.4)** in the lower right cell shows the a normalized ratio between the standard deviation and mean. These three values are shown for NLINFIX (the model benchmark) LINFIX, NLIN, LIN and BESTFIT (the representative industry standard) respectively. The last two columns show the **percentage of solutions where NLINFIX found a better solution than LINFIX (T6.5)**, and the **number of instances (T6.6)** found to be feasible from the set of generated instances. Table 7 shows the information gathered from simulation sorted by the various instances of applications to be allocated. Overall the costs understandably increase as the amount of applications to be allocated increases. Of interest, is how the solutions found by the various models separate in magnitude as the order size increase? Comparing NLINFIX to LINFIX, the average solution gap gradually increases as the number of applications to be allocated (OS) increases. This effect is also present for the percentage of models where the NLINFIX solution was better than the LINFIX solution. A possible explanation for this effect could be that once the allocation base increases, the non-linear models have more room for improvement. NLINFIX and LINFIX show the cost minimal solutions, with only minor separation between them, followed by NLIN and LIN which show solutions with about 20-25% higher costs. Overall, compared to using BESTFIT as an allocation model, if system managers use energy efficient models like NLINFIX or LINFIX, the costs could be reduced by 41%. The same effect can be observed in Table **8** for the instances are grouped by the number of available infrastructure nodes (RC).

Table 7. Cost Evaluation by Number of Applications

PAR	**Ideal**		**2nd Best**		**NoFIX**				**Industry Std.**		**SOL N≻L**	***# Feas***
	(NLINFIX)		***(LINFIX)***		***(NLIN)***		***(LIN)***		***BESTFIT***			
OS-5	773	*(0,546)*	774	+0,2% *(0,544)*	1155	+49,4% *(0,545)*	1165	+50,7% *(0,542)*	1361	+76,1% *(0,543)*	3,7%	*1314*
OS-10	1438	*(0,551)*	1442	+0,3% *(0,549)*	1975	+37,4% *(0,543)*	1984	+38,0% *(0,542)*	2443	+69,9% *(0,469)*	1,8%	*1248*
OS-15	2100	*(0,564)*	2110	+0,5% *(0,561)*	2774	+32,1% *(0,552)*	2782	+32,5% *(0,552)*	3529	+68,1% *(0,471)*	2,2%	*1205*
OS-20	2815	*(0,571)*	2822	+0,2% *(0,568)*	3641	+29,4% *(0,548)*	3648	+29,6% *(0,548)*	4732	+68,1% *(0,469)*	1,4%	*1060*
OS-25	3481	*(0,587)*	3500	+0,6% *(0,580)*	4399	+26,4% *(0,554)*	4403	+26,5% *(0,553)*	5794	+66,5% *(0,463)*	2,8%	*1084*
OS-30	4141	*(0,579)*	4176	+0,8% *(0,573)*	5196	+25,5% *(0,556)*	5200	+25,6% *(0,555)*	7071	+70,7% *(0,448)*	3,5%	*858*
OS-35	4713	*(0,571)*	4736	+0,5% *(0,567)*	5788	+22,8% *(0,538)*	5793	+22,9% *(0,538)*	7924	+68,1% *(0,445)*	2,1%	*951*
OS-40	5306	*(0,594)*	5345	+0,7% *(0,585)*	6514	+22,8% *(0,559)*	6519	+22,9% *(0,559)*	9010	+69,8% *(0,434)*	4,9%	*650*
OS-45	5934	*(0,574)*	5966	+0,5% *(0,567)*	7143	+20,4% *(0,545)*	7146	+20,4% *(0,545)*	9921	+67,2% *(0,422)*	3,7%	*866*
OS-50	6580	*(0,593)*	6622	+0,6% *(0,586)*	7921	+20,4% *(0,561)*	7923	+20,4% *(0,561)*	11275	+71,4% *(0,409)*	4,1%	*491*

Table 8. Cost Evaluation by Number of Nodes

PAR	Ideal		2nd Best		NoFIX				Industry Std.		SOL	#
	(NLINFIX)		*(LINFIX)*		*(NLIN)*		*(LIN)*		*BESTFIT*		N≻L	*Feas*
RC-5	1664	*(0,648)*	1665	+0,0% *(0,648)*	1906	+14,6% *(0,598)*	1910	+14,8% *(0,597)*	2172	+30,5% *(0,608)*	7,1%	*364*
RC-10	2046	*(0,752)*	2049	+0,1% *(0,750)*	2434	+19,0% *(0,677)*	2437	+19,1% *(0,676)*	3019	+47,6% *(0,678)*	2,6%	*500*
RC-15	2933	*(0,737)*	2941	+0,3% *(0,734)*	3517	+19,9% *(0,658)*	3521	+20,0% *(0,6579*	4582	+56,2% *(0,641)*	2,7%	*930*
RC-20	2830	*(0,835)*	2838	+0,3% *(0,832)*	3503	+23,8% *(0,735)*	3508	+24,0% *(0,734)*	4497	+58,9% *(0,724)*	1,5%	*802*
RC-25	3443	*(0,799)*	3459	+0,5% *(0,795)*	4238	+23,1% *(0,720)*	4244	+23,2% *(0,719)*	5723	+66,2% *(0,680)*	2,5%	*1056*
RC-30	3168	*(0,830)*	3177	+0,3% *(0,827)*	3995	+26,1% *(0,750)*	4002	+26,4% *(0,749)*	5290	+67,0% *(0,702)*	1,5%	*1064*
RC-35	3688	*(0,853)*	3712	+0,7% *(0,848)*	4652	+26,2% *(0,777)*	4659	+26,3% *(0,776)*	6259	+69,7% *(0,706)*	2,9%	*1243*
RC-40	3440	*(0,841)*	3460	+0,6% *(0,834)*	4410	+28,2% *(0,781)*	4416	+28,4% *(0,780)*	5968	+73,5% *(0,690)*	2,8%	*1166*
RC-45	3695	*(0,848)*	3727	+0,9% *(0,841)*	4790	+29,6% *(0,785)*	4797	+29,8% *(0,784)*	6607	+78,8% *(0,693)*	3,3%	*1289*
RC-50	3764	*(0,856)*	3791	+0,7% *(0,848)*	4870	+29,4% *(0,800)*	4877	+29,6% *(0,798)*	6739	+79,0% *(0,696)*	3,4%	*1313*

Table 9. Cost Evaluation by Resource Factor

PAR	Ideal		2nd Best		NoFIX				Industry Std.		SOL	#
	(NLINFIX)		*(LINFIX)*		*(NLIN)*		*(LIN)*		*BESTFIT*		N≻L	*Feas*
RF-0,25	1501	(0,619)	1519	+1,2% (0,612)	2090	+39,2% (0,595)	2094	+39,5% (0,593)	3223	+114,8% (0,660)	4,2%	*2928*
RF-0,50	3012	(0,655)	3028	+0,5% (0,650)	3847	+27,7% (0,625)	3850	+27,8% (0,624)	5600	+85,9% (0,631)	2,9%	*2725*
RF-0,75	4509	(0,672)	4525	+0,4% (0,669)	5557	+23,3% (0,637)	5563	+23,4% (0,636)	7171	+59,0% (0,632)	1,9%	*2379*
RF-1,00	5086	(0,730)	5109	+0,5% (0,727)	6138	+20,7% (0,699)	6151	+20,9% (0,697)	7207	+41,7% (0,677)	1,8%	*1695*

Table 10. Cost Evaluation by Allocation Time

PAR	Ideal		2nd Best		NoFIX				Industry Std.		SOL	#
	(NLINFIX)		*(LINFIX)*		*(NLIN)*		*(LIN)*		*BESTFIT*		N≻L	*Feas*
AT-1	2014	(0,823)	2015	+0,4% (0,819)	2502	+24,2% (0,751)	2506	+24,4% (0,749)	3398	+68,7% (0,694)	2,9%	*2170*
AT-2	2451	(0,814)	2464	+0,5% (0,810)	3018	+23,1% (0,743)	3022	+23,3% (0,742)	4124	+68,2% (0,688)	3,7%	*2170*
AT-3	3068	(0,829)	3084	+0,5% (0,823)	3834	+25,0% (0,760)	3840	+25,2% (0,758)	5131	+67,3% (0,705)	3,1%	*2869*
AT-4	3585	(0,835)	3606	+0,6% (0,831)	4547	+26,8% (0,769)	4554	+27,0% (0,768)	6073	+69,4% (0,710)	2,5%	*2909*
AT-5	4161	(0,813)	4181	+0,5% (0,809)	5312	+27,7% (0,755)	5320	+27,9% (0,753)	7104	+70,7% (0,700)	1,9%	*1779*

In Table 9 the cost effect of the resource factor, the relative size of applications in relation to the nodes they are operated on, is shown. If the applications are roughly the same size as the nodes, meaning at most one application can be allocated to a node at a time, the solutions found by all models are closer to each other, than if two or more can fit on each node. Especially if the resource factor nears one, the problems become computationally more intensive for the non-linear solvers reducing their ability to find a good solution in time. For this reason, for the 'hardest' RF in terms of complexity, LINFIX beat NLINFIX in cost efficiency because the non-linear models were simply unable to find better solutions within the allotted CPU time.

In terms of the timeframe over which the application allocation is to take place, the higher the timeframe, the higher the spread between the solutions found by the various heuristics (Table 10). Here, NLINFIX is closely followed by LINFIX in finding the best solutions to the problem.

Based on the above observations, we can see that there is a distinct order of model preference in terms of cost efficiency, with NLINFIX > LINFIX > NLIN > LIN > BESTFIT. However it should be noted that the above observations are only aggregated for models where the nonlinear solvers actually found a better solution within an hour of solver time. In fact, for NLINFIX, only about 2/3 of the MINLP models were actually solved with the branch and bound algorithms of the GAMS/SBB and of these only 4% of these solutions were in fact better than the LINFIX solutions. The same is true for NLIN, but here only about 38% of the models were solved, and only 2% of these were actually better than the LIN variants. As a result, from a stand-alone perspective LINFIX outperforms NLINFIX, not because it is better, but because it is less prone to solver error. This is mostly evident for larger scenarios. For smaller problems, the error diminishes.

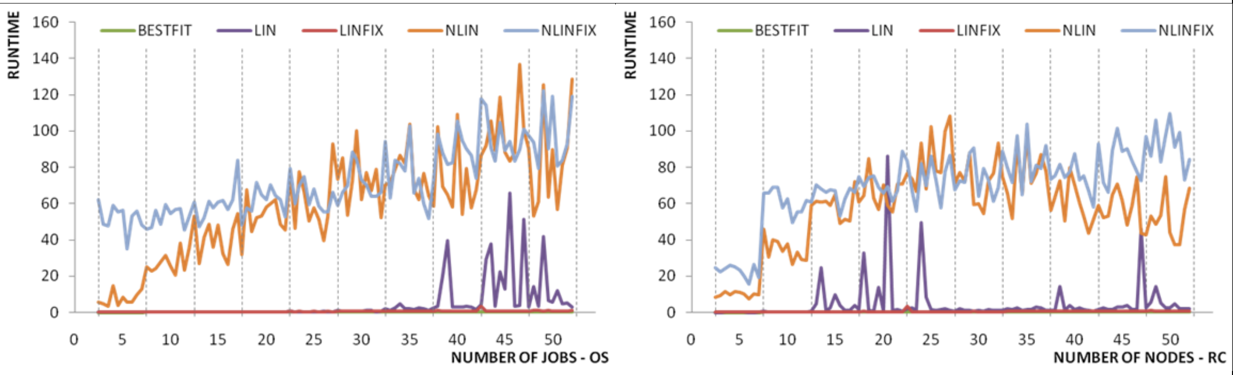
While every solution finding algorithm might eventually succeed to find a solution at some point the time it takes for them to do so decides whether they can be used in practice or not. An algorithm which finds a solution after an hour of work is of no use if the solution is required in ten second intervals; by the time the model has found a solution, the solution may no longer be relevant. Therefore parallel to a cost efficiency analysis, we conduct a runtime analysis of our base model and its cost efficiency and technical model heuristics.

4.3 Runtime Analysis

A runtime analysis can generally be done on two levels: theoretical complexity analysis and the search of what parameters contribute to the "hardness" of a problem. In this work we will do conduct both of these analysis. While the theoretical complexity analysis concluding NLINFIX, LINFIX, NLIN and LIN are NP-Hard can be found in the Appendix A, in the following section we will perform a computational runtime analysis to determine just what parameters make the underlying problems NP-hard. The question now arises, what makes the instances generated above 'hard' or 'easy'.

Figure 3 shows the model runtimes sorted by parameter for the two most representative parameters "applications" and "nodes".

Figure 3. Runtime Evaluations



(a) Runtime Evaluation by Number of Apps *(b) Runtime Evaluation by Number of Nodes*

Figure 3 (a) shows the average CPU time taken against the amount of applications processed. It shows how the CPU time increases as the amount of applications to be scheduled increases for each model. While LINFIX, BESTFIT and (up until order sizes of 30 applications) LIN had runtimes of less than a second, the runtime of NLIN and NLINFIX gradually increased. The same is true for Figure 3 (b) which shows the average CPU time taken against the amount of nodes processed. It shows how the CPU time increases as the amount of nodes to schedule on increases for each model. As the amount of nodes and applications to schedule increases, the solvers take longer to process them. The runtimes of LINFIX and BESTFIT however remain unaffected by this increase.

Since some form of pattern is evident in the runtimes following a *ceteris paribus* change in parameters the dependencies on the respective parameters can be functionally modeled as: $Runtime = \beta_1 * AT + \beta_2 * OS + \beta_3 * RC + \beta_4 * RF + k$.

Following standard practice for multiple regression models, we calculated the $\hat{\beta}$ regression coefficient matrix for each model, with runtime as dependant variable, and AT, OS, RC, RF and *k* (k is a constant) as regresses and yielded the following results shown in Table 11. Each row shows the β coefficients (and their respective t-value of each parameter in parentheses) for each attribute AT, OS, RC and RF respectively. The rightmost column shows the F-values for each function. Since all F's are larger than the critical value 2,80 (for $\alpha = 0{,}01$ with 12236 degrees of freedom) we can deduce that the observed relationship between the dependant variable *Runtime* and the independent regressors $\beta_1, \beta_2, \beta_3$ and β_4 is not merely a random occurrence.

Table 11. Regression Analysis

Model	**AT**	**OS**	**RC**	**RF**	***k***	**F-Value**
NLINFIX	13,018*** *(9,8)*	0,882*** *(9,1)*	0,885*** *(9,4)*	-63,887*** *(-13,0)*	20,794** *(3,1)*	*109,91* ***
NLIN	10,107*** *(9,3)*	1,797*** *(22,7)*	0,619*** *(8,1)*	-45,514*** *(-11,4)*	-13,022** *(-2,4)*	*202,30* ***
LINFIX	0,049 *(1,3)*	0,023*** *(8,5)*	0,015*** *(5,9)*	-0,687** *(-5,1)*	-0,250 *(-1,4)*	*34,65* ***
LIN	0,653 *(0,8)*	0,381*** *(6,4)*	0,008 *(0,2)*	-5,776* *(-1,9)*	-4,492 *(-1,1)*	*11,35* ***
BESTFIT	0,003*** *(4,4)*	0,001*** *(29,4)*	0,001*** *(21,0)*	0,004 *(1,6)*	-0,010** *(-2,5)*	*341,19* ***

*** mark a significance level of more than 99%; ** show a level between 95% and 99% and * indicates a significance of more than 90%.

Looking at the signs, we can conclude, that for all models, increasing the allocation timeframe (AT) results in a direct increase in solver runtime. The same is true for the order size (OS) and for the resource capacity (RC). Finally, the resource factor RF, which controls the general size of the applications simulated, negatively affects the runtime. In words, if the computing tasks are so big that they take up all of the resources of a node on average (i.e. RF $\rightarrow$ 1), solving the allocation problem is quicker than if more than one task can fit onto the system. Comparing effects across models the most prominent difference between the models would be the susceptibility to an increase in solver runtime as the allocation timeframe AT increases for our non-linear model variants. Interestingly, NLIN was more susceptible to a change in OS than NLINFIX, which in turn was more susceptible to an increase in AT than NLIN.

4.4 Bringing it all together: Managerial Implications

While the experimental results above must not be generalized too hastily, they do suggest practical use for the cost efficient management of datacenters. (Dasgupta, A. Sharma, Verma, Neogi, & Kothari, 2011) argue that at a server level, the energy consumed by a server is directly proportionate to its wattage (300W = 300$ per year). BESTFIT, though fast and computationally inexpensive shows weaknesses when faced with heterogeneous computing environments, where the costs generated for systems not equal for all systems. This is where both versions of LIN and NLIN performed well, as they consider these changing costs in their optimization. In comparison LIN and NLIN differ in their need for accuracy. For cases simulated where the NLIN function was quasi-linear (low variance in OLS estimations for the $varco_n$ parameter) the difference between the LIN and NLIN models was so low, that the computational burden to compute the solution for NLIN models makes its slight improvement in energy usage questionable.

The same principle is true for LINFIX and NLINFIX. Both models show great potential in cost reduction and are a must for underutilized datacenter. If technologies are available to reduce the idle costs through PowerNap or comparable features, not including these methods in optimization is costly. Table **8** shows that using NLINFIX rather than BESTFIT results in a cost reduction of 30% for only five servers. Analogous to the monetary worth of this watt savings mentioned by (Dasgupta, A. Sharma, Verma, Neogi, & Kothari, 2011) this results in saving over 500$ per year. For a larger infrastructure of 50 servers this savings potential further increases to 3000$ per year. Once datacenter managers know the costs for each power state of each system, costs for powering and cooling can be cut by up to 43%. These models serve to aid IT Service providers to implement insights gained through our research to allocate their tasks to their resources in a cost-efficient manner, provided their cost situation resembles those presented in this work. In summary, based on the above work we were able to set up the following recommendations based on averages over all simulated data:

Trade-off: Computation Time vs. Solution Quality

Result 1: *If allocation decisions are to be made at regular intervals (less than 1 hour), use LINFIX. This decision will save up to 40,4% energy costs compared to state-of-the-art mechanisms. Compared to the benchmark solution (NLINFIX), LINFIX solutions show a potential loss of up to 2,3%.*

Result 2: *If allocation decisions are less frequent and computation time and infrastructure are not restrictive, we recommend to use NLINFIX. The energy savings potential is 42,5% compared to the state-of-the-art mechanism.*

Result 3: *LIN and NLIN both outperform BESTFIT, however both have a considerably higher computation time and lower energy savings potential than LINFIX. As a result, they are not recommended for implementation in this setting.*

What makes the instances ‘hard’?

Result 4: *Allocation Timeframe (AT) and the Resource Factor (RF) have the largest impact on the computational runtime of NLINFIX and LINFIX.*

5 Conclusions and Outlook

In this work we set out to explore to what extent operation costs of datacenters can be reduced through energy-aware allocations of computational tasks. Understanding how costs are generated, and which factor generates what potential is essential for successful cost reduction strategies. This work contributed to this understanding of costs in datacenters by formulating as well as numerically evaluating various cost models in cost minimization approaches. Through analysis of related work and methods used by datacenters today, we found potential short-term reduction of energy costs through efficient allocation. We found indications of cost saving potential in the choice of allocation mechanism. Based on this finding, we derived and presented a collection of allocation optimizers. Here we argue that there is no single mechanism that satisfies all purposes as each allocation problem has its own set of assumptions and requirements acting on it. For example some allocation procedures need to be allocated within a matter of seconds, while other problems may be given more time for this decision.

Reflecting the distinct requirements of different foci in optimization, a catalogue of co-existing models is needed and thus is provided by this work in a series of model heuristics. We elaborated the models and reflected the unique properties inherent to each version as stipulated by the model requirements. Each model was evaluated and we found that the four models performed significantly better than BESTFIT. This in turn suggests that using our mechanisms, specifically LINFIX, versus currently used platform schedulers could significantly reduce the costs of operation. The trade-off decision between computational speed and allocation efficiency is a challenging choice even for the most experienced project manager. The surprisingly high efficiency of LINFIX as well as the insignificant implementation costs and low computational cost found in this simulation trial makes further research on this path promising. We determined that a good trade-off was achieved while sacrificing only a minor fraction of efficiency. The cost savings potential ranged from 500$ to 3000$ per year, without requiring a large upfront investment. This inclusion could create additional savings opportunities. Energy efficient placement of applications could generate second order effects such as cooling or floor space of datacenters, the operational costs could be lowered even further.

Looking into the future, much work must still be done. In this work, the central focus on energy costs was based on benchmark results focusing only on the CPU system utilization. Little or no insights are gathered into the interplay of memory and CPU. Perhaps by looking at the interplay between hardware within a system, further savings potential could be found. For example, the cooperative effect of a memory intensive task and a CPU intensive task could be investigated. Further, pushing towards regenerative energy usage, the allocation models could include varying power sources when allocating task to nodes. For example, if a datacenter has a pool of solar panels, some applications could be scheduled on solar powered machines when sufficient power is produced.

6 References

Albers, S. (2010). Energy-Efficient Algorithms. *Communications of the ACM*, *53*(5), 86-96. doi: 10.1186/1748-7188-5-11.

Bottcher, J., Drexl, a, Kolisch, R., & Salewski, F. (1999). Project Scheduling Under Partially Renewable Resource Constraints. *Management Science*, *45*(4), 543-559. doi: 10.1287/mnsc.45.4.543.

Burge, J., Ranganathan, Partha, & Wiener, J. L. (2007). Cost-aware Scheduling for Heterogeneous Enterprise Machines (CASH'EM). *IEEE International Conference on Cluster Computing*, 481-487.

Caprara, A., Fischetti, M., & Toth, P. (2002). Modeling and Solving the Train Timetabling Problem. *Operations Research*, *50*(5), 851-861. doi: 10.1287/opre.50.5.851.362.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., & Doyle, R. P. (2001). Managing Enery and Server Resources in Hosting Centers. *ACM SIGOPS Operating Systems Review*, *35*(5), 103-116. Alberta, Canada.

Chen, G., Malkowski, K., Kandemir, M., & Raghavan, S. (2005). Reducing power with Performance Constraints for Parallel Sparse Applications. *Proceedings of the 19th International Symposium on Parallel and Distributed Processing* (p. 8pp).

Dasgupta, G., Sharma, A., Verma, A., Neogi, A., & Kothari, R. (2011). Workload Management for Power Efficiency in Virtualized Data Centers. *Communications of the ACM*, *54*(7), 131-141.

Feitelson, D. G. (2002). Workload Modeling for Performance Evaluation of Complex Systems: Techniques and Tools. *Lecture Notes in Computer Science*, *2459*, 114-141. Springer Verlag.

Freeh, V. W., Lowenthal, D. K., Pan, F., Kappiah, N., Springer, R., & Rountree, B. L. (2007). Analyzing the Energy-Time Trade-Off in High Performance Computing Applications. *IEEE Transactions on Parallel Distribution Systems*, *18*(6), 835-848.

Hamann, H. F. (2008). A Measurement-Based Method for Improving Data Center Energy Efficiency. *Proceedings of the IEEE International Conference on Sensor Networks*.

Mastroleon, L., Bambos, N., Kozyrakis, C., & Economou, D. (2005). Automatic power management schemes for Internet servers and data centers. *Global Telecommunications Conference, 2005. GLOBECOM 05, IEEE*.

Moore, J., Chase, J., Ranganathan, P, & Sharma, R. K. (2005). Making scheduling “cool”: temperature-aware workload placement in data centers. *Proceedings of the annual conference on USENIX Annual Technical Conference, USENIX Association, Anaheim, CA*.

Nathuji, R., Isci, C., Gorbatov, E., & Schwan, K. (2008). Providing platform heterogeneity-awareness for data center power management. *Cluster Computing*, *11*(3), 259-271.

Park, I., & Pu, I. (2007). Energy Efficient Expanding Ring Search. *Proceedings of the First Asia International Conference on Modelling & Simulation, IEEE Computer Society*.

Pinheiro, E., Bianchini, R., Carrera, E., & Heath, T. (2003). Dynamic cluster reconfiguration for power and performance. *Compilers and operating systems for low power, Kluwer Academic Publishers*, 75-93.

Pisharath, J., Choudhary, A., & Kandemir, M. (2004). Energy management schemes for memory-resident database systems. *Proceedings of the thirteenth ACM international conference on Information and knowledge management, ACM, Washington, D.C., USA*.

Raghavendra, P. (2008). Optimal algorithms and inapproximability results for every CSP. *Proceedings of the 40th annual ACM symposium on Theory of computing, ACM, Victoria, British Columbia, Canada*.

Rivoire, S., Shah, M., Ranganathan, Partha, & Kozyrakis, C. (2007). JouleSort: a balanced energy-efficiency benchmark. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, Beijing, China*.

See, S. (2008). *Is there a pathway to a Green Grid*.

Son, S., Malkowski, K., Chen, G., Kandemir, M., & Raghavan, P. (2007). Reducing energy consumption of parallel sparse matrix applications through integrated link/CPU voltage scaling. *Journal of Supercomputing*, *41*(3), 179-213.

Vengerov, D. (2009). A reinforcement learning framework for utility-based scheduling in resource-constrained systems. *Future Generation Computer Systems*, *25*(7), 728-736. doi: 10.1016/j.future.2008.02.006.

Wang, X., & Chen, M. (2008). Adaptive power control for server clusters. *2008 IEEE International Symposium on Parallel and Distributed Processing*, 1-5. Ieee. doi: 10.1109/IPDPS.2008.4536425.

Weiser, M., Welch, B., Demers, A., & Shenker, S. (1994). Scheduling for reduced CPU energy. *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*. USENIX Association, Monterey, California.

Yao, F., Demers, a, & Shenker, S. (1995). A scheduling model for reduced CPU energy. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 374-382. IEEE Comput. Soc. Press. doi: 10.1109/SFCS.1995.492493.

APPENDIX A

6.1 Computational Complexity Analysis

Proposition A.1: LIN is NP-hard.

Proof: We reduce the 0-1 Multiple Knapsack Problem (MKP), which is NP-hard (Martello & Toth, 1990) to LIN. The MKP is defined as:

$max_X\, K = max \sum_n^N \sum_j^J p_j X_{nj}$ $\qquad \forall\, j \in J, \forall\, n \in N,$ (MKP1)

Subject to:

$X_{nj} \in \{0,1\},$ $\qquad \forall\, j \in J, \forall\, n \in N,$ (MKP2)

$\sum_n^N X_{nj} = 1$ $\qquad \forall\, j \in J$ (MKP3)

$\sum_j^J x_{nj} w_j \leq c_n,$ $\qquad \forall\, n \in N$ (MKP4)

$w_j \geq 0$ $\qquad \forall\, j \in J$ (MKP5)

Any instance of 0-1 MKP can be polynomially transformed into an equivalent instance of LIN by transforming the objective function into a minimization function $min \sum_n^N \sum_j^J (-p_j) X_{nj}$, and by setting T={1} (when reducing LIN to a problem over a single time unit, the index t can be removed from the decision variables; (MKP4) is then modeled through (L2)), $\frac{\overline{b_n}}{cs_n} = 1\ (n \in N), cd_j = -p_j\ \ (j \in J), cs_n = \sum_j^J cd_j\ (n \in N), md_j = w_j\ (j \in J), \overline{ms_n} = c_n\ \ (n \in N).$ □

Proposition A.2: LINFIX, NLIN and NLINFIX are NP-hard

Proof: For LINFIX, setting $kfix_n$= 0 for all *N*, we yield an instance of LIN, which is NP-hard (see proof above). Likewise, setting a_n = 0, c_n = 1, for all *N*, in NLIN, yields an instance of LIN. Finally, setting a_n = $kfix_n$ = 0, c_n = 1, in NLINFIX resembles an instance of LIN, which is NP-hard (as above). □