Smart Grid-aware Scheduling in Data Centres

Markus Mäsker[a,*], Lars Nagel[a], André Brinkmann[a], Foad Lotfifar[b], Matthew Johnson[b]

[a] *Zentrum für Datenverarbeitung, Johannes Gutenberg-Universität Mainz*
[b] *School of Engineering and Computing Sciences, Durham University*

Abstract

In several countries the expansion and establishment of renewable energies result in widely scattered and often weather-dependent energy production, decoupled from energy demand. Large, fossil-fuelled power plants are gradually replaced by many small power stations that transform wind, solar and water power into electrical power. This leads to changes in the historically evolved power grid that favours top-down energy distribution from a backbone of large power plants to widespread consumers. Now, with the increase of energy production in lower layers of the grid, there is also a bottom-up flow of the grid infrastructure compromising its stability. In order to locally adapt the energy demand to the production, some countries have started to establish Smart Grids to incentivise customers to consume energy when it is generated.

This paper investigates how data centres can benefit from variable energy prices in Smart Grids. In view of their low average utilisation, data centre providers can schedule the workload dependent on the energy price. We consider a scenario for a data centre in Paderborn, Germany, hosting a large share of interruptible and migratable computing jobs. We suggest and compare two scheduling strategies for minimising energy costs. The first one merely uses current values from the Smart Meter to place the jobs, while the other one also estimates the future energy price in the grid based on weather forecasts. In

*Corresponding author
Email addresses: maesker@uni-mainz.de (Markus Mäsker), nagell@uni-mainz.de (Lars Nagel), brinkman@uni-mainz.de (André Brinkmann), foad.lotfifar@durham.ac.uk (Foad Lotfifar), matthew.johnson2@durham.ac.uk (Matthew Johnson)

spite of the complexity of the prediction problem and the inaccuracy of the weather data, both strategies perform well and have a strong positive effect on the utilisation of renewable energy and on the reduction of energy costs.

This work improves and extends the paper of the same title published on the *SustainIT* conference [1]. While that paper puts more emphasis on the utilisation of green energy, the new algorithms find a better balance between energy costs and turnaround time. We slightly alter the scenario using a more realistic multi-queue batch system and improve the scheduling algorithms which can be tuned to prioritise turnaround time or green energy utilisation.

Keywords: Smart Grid, scheduling, energy efficiency

---

1. Introduction

With the "Energiewende" (energy transition) [2], the German government decided to enforce a more sustainable energy development policy improving the overall energy efficiency and the share of renewable energy. Many other countries follow similar policies. The reasons are manifold and include the reduction of greenhouse gas emissions, the risk of nuclear accidents and the costs of and the dependency on fossil fuels. The shift from nuclear and coal-fired power plants towards wind and solar power plants results in a widespread energy generation in subgrids and in the decoupling of energy production and energy consumption. Therefore, the following problems need to be addressed:

1. Energy generation within distribution grids makes the grid more complex and causes problems because many of the transformers to the respective transmission grids are often not capable of transporting the (peak) energy produced by windmills and solar collectors. Since it would be costly to purchase high-performance transformers to adapt the grid to accommodate the mini power plants, it is desirable to consume energy locally when it is produced.

2. Energy is not always produced when it is needed and not always needed

when it is produced. For this reason, it is desirable to adapt consumption to generation as long as there is no efficient way of storing energy.

3. Energy consumption should be contained since energy that is not used does not have to be generated.

Several countries try to overcome the first two problems by introducing a *Smart Grid* that monitors the state and the load flow of the electrical grid's elements and provides these data in real time such that measures can be taken if necessary. Stimuli for consumers are prices that reflect the situation in the grid. If too much energy is produced in an area, the price will drop for the consumers there.

In anticipation of the Smart Grid and dynamic energy prices, solutions for these problems were developed in the project "GreenPAD"[1] which considered a scenario in Paderborn, Germany. It focused on a local data centre offering computing services to research institutes and small companies. The main issue was to build a Green Control Centre that schedules the incoming workload to time periods of energy surplus and thus lower energy prices.

In this paper we describe the scenario and challenges and evaluate two types of schedulers using different performance metrics. To keep the expenses low, we concentrate on schedulers that use either free or inexpensive data. Aside from standard schedulers and optimised schedulers with perfect knowledge that are run for comparison, the schedulers utilise data from the Smart Grid including information about the current local energy production and consumption. The so-called *green scheduler* also analyses low-cost weather recordings and forecasts to predict the future energy surpluses and prices.

Although this task is in principle more complex than the prediction of on-site solar and wind power plants (treated e.g. in [3, 4, 5, 6]), we show that our low-cost schedulers already suffice to increase the share of renewable energy to a nearly optimal value. The price to pay is an increase in the turnaround time

[1] http://www.green-pad.de

so that one has to make a compromise between the green energy rate and the service quality. In terms of our assumed price model, the energy costs saved amount to about 7.7 % in a scenario where 9.9 % would be optimal. From these results it follows that, at least for small data centres, the purchase of more expensive weather or energy forecasts would not be profitable as they might not save the money they cost.

This work improves and extends the paper of the same title published on the *SustainIT* conference [1]. While the previous paper puts more emphasis on the utilisation of green energy, the improved algorithms find a better balance between energy costs and turnaround time. We slightly alter the scenario using a more realistic multi-queue batch system and improve the scheduling algorithms which can be tuned to prioritise turnaround time or green energy utilisation. For the evaluation of the algorithms we include a new cluster trace with a completely different load profile and investigate the algorithms' performance for non-preemptive job scheduling.

The paper is structured as follows: After a discussion of related work, we describe the scenario in more detail. In Section 2 we outline the software consisting of a scheduler and an energy prediction component, where the latter is only used by the energy-efficient scheduler. The schedulers are compared and evaluated in Section 3 before the paper is summarised and concluded in Section 4.

1.1. Related Work

The deployment of renewable energy has recently gained popularity in the IT industry [7, 8] and inspired projects in both, academia and industry, for example *DC4Cities*[2], *Parasol*[3], *GreenStar Network*[4], *GreenQloud*[5] and *Green Mountain*[6]. The main research challenge is the irregular power output of wind

[2] http://www.dc4cities.eu
[3] http://parasol.cs.rutgers.edu
[4] http://www.greenstarnetwork.com
[5] https://www.greenqloud.com
[6] http://www.greenmountain.no

farms and solar collectors. Solutions to these problems usually include one or more of the five key aspects that were defined by Deng et al. [8]: 1. generation models, 2. prediction of renewable energy, 3. capacity planning, 4. scheduling within and 5. in between data centres. In this paper we concentrate mostly on the fourth point, but also consider the second one. Therefore, this survey first discusses publications related to energy prediction and then work about energy-aware scheduling.

Improvements in *numerical weather prediction* (NWP) and in power forecast algorithms have considerably improved the accuracy of the forecast models in the past decades [9]. The taxonomy of forecast models is so diverse that we cannot cover it completely, but only name a few models: direct time series forecasting (e.g. [10, 11]), time series models in combination with neural networks ([12, 13]), direct power forecast models with statistical improvements (e.g. [14, 13]), models dealing with non-linear power curves and the accuracy of NWP input (e.g. [9, 15]). For an extensive survey the reader is referred to Giebel et al. [15].

Complementing the energy prediction, it is also useful to predict how much energy will be consumed. Like in our case, this can be difficult because of missing data or fluctuations in the consumers' behaviour. These fluctuations are also a problem for providers who need to adapt the production to the consumption or, especially in Smart Grids, the consumption to the production by offering incentives [16, 17].

In [18] Brown and Renau introduce *ReRack*, a simulation environment for analysing the costs associated with the employment of renewable energy. The software includes an *optimizer* that uses a genetic algorithm to improve the system subject to a user-defined cost function. The paper suggests input and algorithms in the form of models and parameters, but only provides a very brief section about the actual application of the tool. Ren et al. present a framework in [19] that helps to reduce a data centre's energy costs and possibly its carbon footprint. The analysis uses linear programming and is based on the energy prices for on-site and off-site green energy as well as energy from other sources. Provided that the carbon footprint target is not too high, they show that the

on-site generation of green energy can also reduce the costs. Since we do not consider on-site energy production, we cannot apply their framework.

The following papers discuss energy-aware schedulers and present software solutions for different scenarios involving data centres supplied by small on-site power plants. While their use cases differ from ours, the basic idea is similar as they want to execute jobs when (local) green energy is available. *SolarCore* [3] considers a system that relies on solar power as the main energy source, but automatically switches to grid power when solar energy drops below a threshold. By controlling the power state of servers, a green energy utilisation of 82% is achieved with little impact on performance. Solely relying on renewable sources, *Blink* [20] puts servers in active or inactive mode depending on the energy situation. Its major drawback is that using only renewable sources is unrealistic and causes unbounded performance degradation. *iSwitch* [4] explores a design that puts servers into two groups: the first half is supplied with energy from the grid, the other half with on-site wind energy. Based on the availability of wind energy, *iSwitch* migrates load between the groups. The system introduced in [21] is a real-time scheduler for batch and service jobs based on off-site solar and wind energy production and they use short-term weather forecasts to get more precise energy predictions. The project *Parasol* at Rutgers University proposes the software systems *GreenSlot* [5] and *GreenHadoop* [6]. *GreenSlot* is a batch job scheduler for data centres which are powered by an on-site photovoltaic array and use the electrical grid only as a backup. The scheduler predicts the solar energy available and places the jobs in such a way that their deadlines are met and that the utilisation of green energy is maximised. *GreenHadoop* is a similar system designed for Hadoop jobs. By deferring the map and reduce jobs, it tries to match the variable green energy supply.

Besides placing and migrating jobs within a single data centre (which is also what we do), many papers consider the case of migrating jobs between geographically dispersed data centres. *GreenWare*, proposed by Zhang et al. [22], is a middleware that dispatches jobs to data centres based on local energy prices. The authors found that, if energy is dynamically priced based on the proportion

of fossil energy, the usage of fossil energy can be significantly reduced. *Free Lunch* [23] co-locates data centres with renewable energy generation sites and migrates workload between data centres according to available power. *GreenNebula* [24], developed by the *Parasol* project, follows a similar approach. It extends the *OpenNebula* cloud manager and maximises the use of green energy by migrating VMs across data centres. In [25] Li et al. assume a dynamic pricing market and propose a collaboration framework for energy cost optimisation that couples data centres with the electricity market. They claim that this collaboration can reduce the costs by up to 75%.

To the best of our knowledge there are no papers that schedule jobs based on data from Smart Grids. However, there are papers that consider scheduling with respect to dynamic energy prices. Niehörster et al. [26] propose a scheduling mechanism for a dynamic pricing model based on the spot market of the *European Energy Exchange*[7] (EEX). A multi-agent system, which is aware of the price, is placed on top of a cloud's infrastructure layer. Scheduler agents collaborate with worker agents that monitor the jobs during their execution and control the system such that it fulfils the service-level agreements while minimising the electricity costs.

1.2. Scenario

This paper describes schedulers that increase the usage of locally produced energy and thereby reduce the energy costs, but in contrast to most related work, a more complex scenario is considered. The energy price depends on the surplus in the local grid so that energy predictions are only useful if they are made for the whole local grid involving suppliers and consumers, and not only for an on-site power plant. This section describes the scenario in more detail.

1.2.1. Smart Grid

The hypothetical Smart Grid that we consider in our experiments is located in Paderborn, Germany. Aside from the data centre, the local electrical grid

[7] http://www.eex.com

supplies companies of different size and residential areas. As depicted in Figure 1, the suppliers in this medium voltage (MV) grid are wind farms and photovoltaic collectors on rooftops. The fossil-fuelled power plants are located in the extra high voltage (EHV) grid outside of the MV grid. Although the Smart Grid is not yet in place, the necessary values are made available by the local grid provider *Westfalen Weser Energy*[8] (WWE). Besides the data centre's usage, data from additional metering points are supplied which are used to train the linear models for energy prediction (cf. Section 2.1). These meterings are the energy flow between the MV grid and the high voltage (HV) grid, the energy generated by the wind farms and the contribution of the solar panels. However, since the panels are located in residential areas, these last values are actually the difference between production and consumption in the respective low voltage (LV) grids.

1.2.2. Weather data for energy prediction

The green schedulers include planning algorithms that predict the local energy production in the near future. Apart from the energy values, these schedulers require current weather readings and forecasts. The former are taken from the closest weather station of the *German Weather Service*[9] (DWD) in Bad Lippspringe, the latter from the *European Weather Consult*[10] (EWC). Instead of predicting the energy oneself, one could also purchase energy forecasts, but these forecasts are usually only offered to energy companies, cover wider areas and can be very expensive. The energy prediction models will be described in Section 2.1.

1.2.3. Workload

The share of energy consumed from the HV grid can be reduced because of two reasons: First, computing clusters or clouds are usually not fully utilised

[8] http://ww-energie.com
[9] http://www.dwd.de
[10] http://www.weather-consult.com

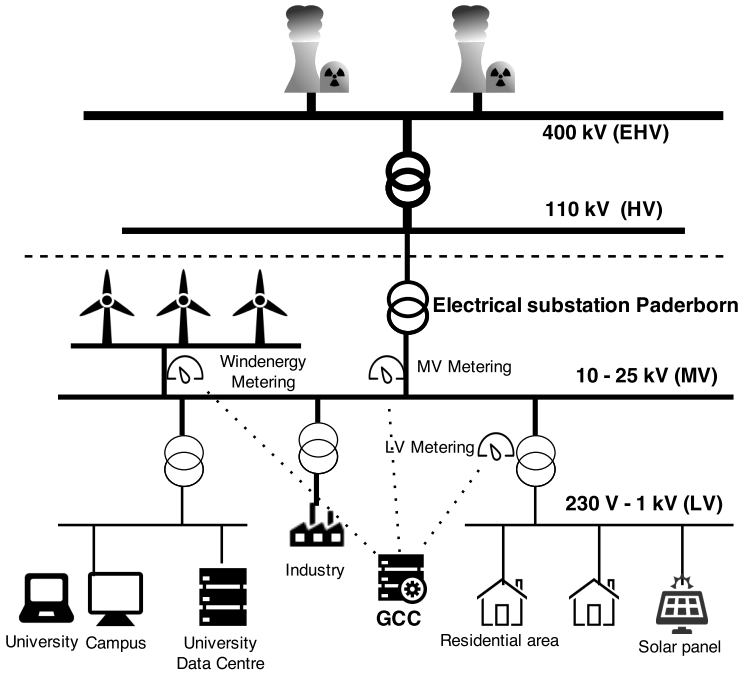


Figure 1: Proposed smart grid architecture in Paderborn

so there is the possibility of running jobs at more favourable times. Second, we are interested in data centres with a large share of interruptible and migratable computing jobs, usually so-called batch jobs that can be stopped and restarted as well as replaced essentially at any time. In our experiments, a large data centre is simulated using freely available traces[11].

1.2.4. Objective target

The goal is to increase the share of locally produced green energy while keeping the performance degradation low and without compromising the average throughput of the data centre. By delaying and interrupting jobs, however, it is obvious that the quality of service will degrade. We use a natural quality measure for batch jobs, namely the *average turnaround time (TAT)*. The turnaround time of a job is the time it stays in the system, i.e. the time from its arrival at

[11] http://www.cs.huji.ac.il/labs/parallel/workload

the queue to its completion.

Besides the share of renewable energy we will also assess the schedules by calculating the energy costs. Since there is no suitable price model in Germany yet, we use a hypothetical one. Assuming that in future the energy price will be dynamic and highly dependent on where the energy is produced, this model sets the price according to the current surplus. In Section 3.1 we will describe it in more detail.

2. Green Control Centre

The *Green Control Centre* is our implementation of the energy-efficient cloud environment. As depicted in Figure 2, it consists of the *Energy Prediction Component* and the *Scheduler Component* that will be described in the following subsections. The Green Control Centre was embedded in an *OpenStack*[12] cloud environment, for which reason Figure 2 displays a few OpenStack components. Nevertheless, the concept is generally valid for any data centre running computationally intensive jobs.

2.1. Energy Prediction Component

The *Energy Prediction Component* predicts the future availability of renewable energy. Its inputs are the current energy and weather readings as well as the weather forecast. The subcomponents *Wind Model* and *Photovoltaic Model* use these inputs to compute the energy forecast for the wind farms and the low voltage grids (with their solar collectors), respectively. The *Grid Model* combines the output of these two subcomponents with a consumption estimate by the *Consumer Model* to predict the energy surplus or shortage in the Paderborn grid. This difference between production and consumption is the value that determines the energy price in our scenario. It must be evened out by either receiving energy from or providing energy to the high voltage grid. In the worst

[12] http://www.openstack.org/

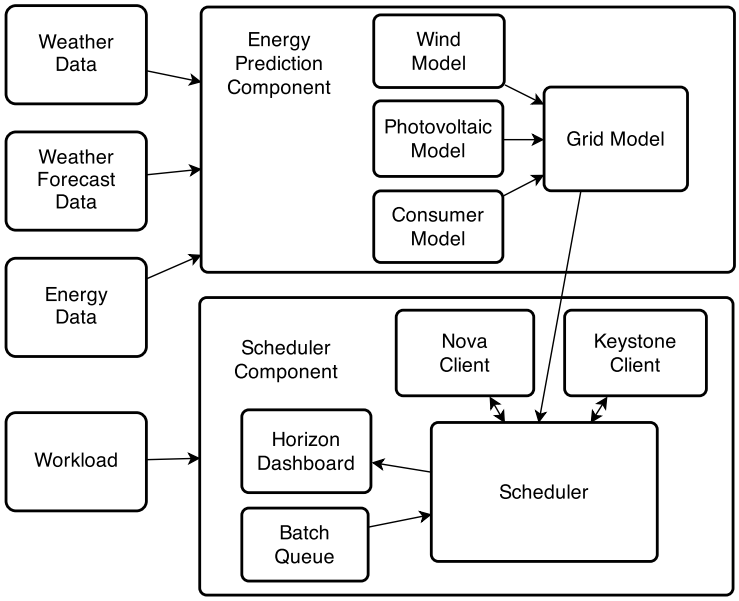


Figure 2: Green Control Centre architecture diagram

case, the energy suppliers have to be turned off while the grid provider still has to pay for them.

2.1.1. Wind Energy Prediction

To build a model that predicts the wind energy based on weather forecasts, it is necessary to determine the relevant weather attributes. This is done by computing the Pearson correlation of individual attributes with the generated wind energy. The results are displayed in Table 1. The selected attributes are *wind speed*, *wind direction*, *temperature* and *atmospheric pressure*. Unsurprisingly, the wind speed shows the strongest correlation. Yet, although the value of 0.678 indicates a *high correlation*, it should be even higher. The reason for the relatively low values is that the quality of the weather data is affected by the distance between the weather station and the wind park (17 km) and by the fact that European weather services measure the wind speed at a height of 10 metres while the hubs of the wind turbines are between 36 and 62 metres high. We can quantify the error because we also have the wind speed readings the turbines

take at hub height for operational purposes. Their correlation coefficients are around 0.82. The discrepancy in the coefficients suggests a significant difference between the weather readings and the actual weather at the wind farms.

Our analysis of the wind speed forecast data shows that the expected error grows with the wind speed. Although the overall expected error of 1.0 ms^{-1} is quite acceptable, it becomes large for high wind speeds, for instance 5.3 ms^{-1} for a wind speed of 10.0 ms^{-1}. Translated into wind energy, the values measured range between 0 and 31 MW. The average error of our wind energy prediction remains high due to its immediate dependence on the weather forecasts. It amounts to 3.0 MW.

For the prediction of wind energy we applied linear regression. The linear model was trained with the chosen weather attributes taking the readings of exactly one year. Once a model is trained, the wind energy can be determined by feeding the weather forecast attributes to it. The complexity for creating a model is linear in the number of training values so that the computation does not take a lot of time. However, since the model does not have to be renewed very often and since this can be done in a separate process, the scheduler should not be slowed down by it anyway. The computation of one energy value is linear in the number of attributes and therefore very fast.

We investigated whether more advanced machine learning techniques would improve the forecast quality, but did not see a significant change. One approach was the extension of the linear model by clustering the weather data before applying linear regression. Another approach tested was the popular power curve model (e.g. [9], [15]) which directly maps the wind speed to the generated energy. Based on the weather readings, we derived power curves for the wind farms and used them for the prediction, but we could not see any improvement. We suppose that the prediction could only be improved if the weather measurements were better. However, the evaluation in Section 3 will show that the quality is already sufficient for our purposes.

Table 1: Pearson correlation of weather measurements and wind energy production

	Correlation		
Weather attribute	Wind farm 1	Wind farm 2	Wind farm 3
Wind speed	0.678	0.622	0.591
Wind direction x-axis	0.093	0.137	0.097
Wind direction y-axis	0.305	0.151	0.056
Sunshine	-0.172	-0.133	-0.108
Temperature	-0.181	-0.164	-0.140
Atmospheric pressure	-0.229	-0.249	-0.234
Rain	0.118	0.083	0.093

2.1.2. Photovoltaic Energy Prediction

The photovoltaic energy prediction is more complicated in our scenario because we do not have exact measurements for the installed panels. Instead, we have the energy exchange of one of the LV grids with Paderborn's MV grid. The LV grid includes not only the production of the panels, but also the consumption of the respective residential area. Additionally, since the installed photovoltaic power of the whole grid (3.5 MW peak) is about ten times larger than the installed power for the monitored grid (330 kW peak), we have to extrapolate the measurements of the LV grid accordingly.

We determine the significant weather attributes using correlation and apply linear regression to estimate the energy. Table 2 shows that the two relevant attributes are cloud coverage and temperature. Yet in this case, further attributes are reasonable: irradiation angle of the sun, day of the week and time. The latter two are required so that the system can learn the behaviour of the consumers in the LV grid which is assumed to be day-of-the-week and time-of-the-day dependent. The irradiation angle and the cloud coverage determine the amount of solar energy reaching the Earth's surface. For the calculation of the irradiation

Table 2: Pearson correlation of the weather attributes with the low voltage grid energy exchange

Weather attribute	Correlation
Wind speed	0.18967
Wind direction X-Axis	0.02072
Wind direction Y-Axis	0.05713
Cloud coverage	0.67093
Temperature	0.44410
Atmospheric pressure	0.01274
Rain	-0.07279

angle, we use the tool *Pysolar*[13].

The analysis of the cloud coverage forecast quality reveals an average error of 28.6%. It is respectively higher (46.3%) or lower (21.2%) if a cloudless or overcast sky is predicted. The measured energy values of the low voltage grid cover a range of 322 kW surplus and 122 kW demand where the periods of energy demand are almost always at night. At these times, the values show little variance in the consumer behaviour and can be predicted with an average error of less than 20 kW. Surplus situations, on the other hand, have an average error of 50 to 85 kW.

2.1.3. Energy Consumption Prediction

For reasons of privacy, individual energy consumption readings necessary for pattern matching are not available. Yet, even though the precision of the prediction is limited, one can still get a fair estimate by training a linear model using time data like the day of the week, date and time of the day. This allows to roughly predict the general consumption during the day, week or season. In a future Smart Grid, the grid provider could publish anonymised or generalised usage statistics allowing a more elaborate prediction of the consumption.

[13] http://pysolar.org

2.1.4. Prediction of the Grid Exchange

The grid exchange estimated by the *Grid Model* is the amount of energy that has to flow to balance the surplus or shortage in the local grid. For the prediction the Grid Model simply sums up the outputs of the other models: Wind Model, Photovoltaic Model and Consumer Model.

The measured grid exchange values range between 30.6 MW surplus and 21.8 MW demand. The average error of the prediction grows with the green energy surplus and is between 2.5 and 10 MW. We believe that the rather poor precision could be improved by using better weather data, for instance provided by on-site weather stations.

In our scenario we use the grid exchange to derive an energy price that would be provided by a Smart Meter. Since the data centre is integrated into the Smart Grid, we conclude that it has access to the Smart Meter and that the energy price is available at runtime so that it can be used to correct the forecast.

2.2. Scheduler

In the reference implementation (Figure 2), the *Scheduler Component* is embedded into an OpenStack environment where it functions as an energy-aware batch-processing system. The main components are the *Batch Queue* managing the incoming batch jobs and the *Scheduler* generating a schedule and placing the jobs accordingly. Three further components are needed to integrate the service into OpenStack: The *Nova Client* is used to keep track of the cloud infrastructure, to monitor the state of the cloud, i.e. the available resources, and to start and stop virtual machines. *Keystone*, accessed by the *Keystone Client*, is OpenStack's identity service and handles the user and project management. *Horizon* finally is a web-based graphical user interface into which we integrated a read-only view of the batch processing system.

The system creates a home directory and an input and output subdirectory for every cloud user. New jobs and any input parameters are placed in the input directory, the results in the output directory. The home directory is mounted

into the job's virtual machine by a *cloud-init*[14] script .

In the following we describe different implementations of the *Scheduler* subcomponent, namely *FIFO scheduler*, *MATH scheduler*, *green scheduler* (GREEN), *enhanced green scheduler* (ENHG), *cost optimal scheduler* (OPT), *shortest remaining time scheduler* (SRT) and *shortest job first scheduler* (SJF) where the last four schedulers are in some way optimised using additional data and are mainly run for comparison. All schedulers differ in the way in which they choose the times at which the jobs are run. Once this is decided, the placement of the virtual machines is done in a best-fit manner. In doing so, it bypasses the OpenStack scheduling mechanism and directly accesses the administrator interface, transparent to the cloud infrastructure.

Common scheduling objectives in batch processing systems are maximising throughput, minimising the average turnaround time, ensuring fairness and avoiding starvation of jobs. The shortest remaining time scheduler, for example, minimises the average waiting time, but favours short jobs over longer ones. This violates the fairness objective and might lead to job starvation. Finding a suitable trade-off among all these objectives is especially difficult in our case: Minimising energy consumption during phases of high energy costs requires that jobs are deferred, which essentially contradicts almost all mentioned objectives.

The traces used for our evaluation specify neither priorities nor deadlines so that we assign the same priority to every job and set a maximum waiting time. If a job has waited for four days, it will be processed as soon as possible. The batch processing system is inspired by the procedures in place at the HPC system *Mogon* at Mainz University[15]. When a user submits a job, he has to specify the allocation time necessary to run their jobs. The allocation time and the resources requested determine in which of the following queues the job is placed:

- tiny: allocation time of at most one hour

[14] `https://launchpad.net/ubuntu/+source/cloud-init`

[15] `https://hpc.uni-mainz.de/`

- short: allocation time of at most six hours, at most one node
- short node: allocation time of at most six hours, multiple nodes
- long: allocation time of more than six hours, one node
- long node: allocation time of more than six hours, multiple nodes

In order to avoid extreme waiting times, the total workload enqueued must be limited. The limit could be set by the site administrator and should depend on the processing capacity of the data center and the maximum waiting time.

The scheduling process consists of a main phase running a custom scheduler enclosed by a preprocessing and a postprocessing phase. In the main phase, the custom scheduler has to decide whether to run jobs or not. If so, jobs are fetched from the five queues in a round-robin fashion starting with the queue containing the oldest job. Internally, each of the queues is strictly FIFO. If a queue is empty or if the first job cannot be scheduled, the queue is skipped in that round.

The preprocessing phase schedules jobs that must be run regardless of the custom scheduler strategy. In our case this affects jobs having passed their maximum waiting time and – once they have been started – non-preemptive jobs. Again, preprocessing starts with the oldest job and continues in a round-robin fashion. In the non-preemptive case the starvation of a large job is possible when the cluster is repeatedly blocked by smaller jobs. For this reason we apply the following procedure: When the first job of any queue cannot be placed after twice its maximum waiting time, then a large enough share of the cluster is reserved for this job and no other job is allowed to be scheduled there. Generally, the preprocessing phase can be extended by additional rules, for example, concerning hard deadlines and high priority jobs.

During the postprocessing phase, additional jobs from the tiny queue, usually short test runs, are scheduled regardless of the energy costs because a fast response time is desirable. Due to the limited runtime, these jobs have a negligible impact even if no green energy is available. Note that they are only run if the scheduler decides to postpone processing because otherwise the tiny

jobs would have been already scheduled in the main phase.

2.2.1. First In - First Out (FIFO) Scheduler

The FIFO scheduler is a fair scheduler ignorant of the Smart Grid that does not try to utilise green or locally produced energy. It processes the jobs of every queue in the order of their arrival and starts each job in the earliest time slot possible. This is a plausible strategy in batch processing systems that consider neither deadlines nor priorities. It achieves high throughput and fairness.

2.2.2. Shortest Remaining Time and Shortest Job First Scheduler

Since multi-processor scheduling is known to be an NP-hard problem [27], there is no known efficient algorithm for computing a turnaround time-optimal schedule. We use the shortest remaining time scheduler (SRT) when all jobs are preemptive and the shortest job first scheduler (SJF) when all jobs are non-preemptive. Both schedulers are proven to be optimal for single-processor scheduling with respect to the average turnaround time [28].

2.2.3. MATH Scheduler

The MATH scheduler memorises former energy prices and former sizes of the five queues and uses these data to assess the current situation. The pseudo code of the scheduler is given in Algorithm 1 and Algorithm 2. The former creates a local view, the latter a global view of prices and queues. They are used to decide whether to place new jobs or to wait.

In each round, the algorithm first computes the total remaining allocation time of all jobs (line 1). This value denotes the enqueued workload size. For the local view, it keeps two lists of fixed length n, L_P and L_Q, which respectively store the energy prices and workload sizes of the last n hours. In line 4 and 5, the rank of the current price and queue size are determined by sorting the lists in ascending order. If $\texttt{index}_Q$, the position in L_Q, is not smaller than $\texttt{index}_P$, the position in L_P (line 6), jobs are placed subject to the available resources. As described is Section 2.2, the algorithm iterates through the five queues in a round robin fashion (line 7 - 10).

Algorithm 1 Specific code of MATH scheduler

1: `remAllocationtime` $\leftarrow \sum_{q \in queues} \sum_{j \in q} remainingAllocationtime(j)$
2: update L_P by replacing the oldest entry with the current price p_c
3: update L_Q by replacing the oldest entry with the current workload size
4: $\texttt{index}_P \leftarrow$ index of current price in L_P (sorted in ascending order)
5: $\texttt{index}_Q \leftarrow$ index of current workload size in L_Q (sorted in ascending order)
6: **if** $\texttt{index}_Q \geq \texttt{index}_P$ **or** `checkGlobalStatus`(remAllocationtime, p_c) **then**
7: **for all** job $\in$ queues **do**
8: // round-robin queueing
9: **if** enough resources available **then**
10: dequeue job and process it

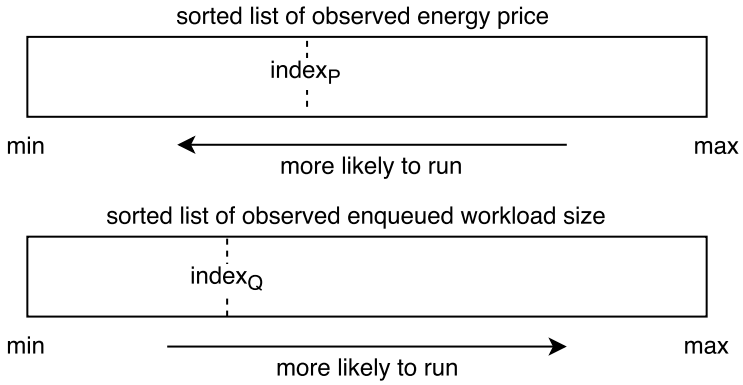


Figure 3: Graphical representation of the mathematical model

A small $\texttt{index}_P$ ($\texttt{index}_Q$) implies that the current energy price (or workload size) is low compared to recent prices (or sizes) while a large $\texttt{index}_P$ ($\texttt{index}_Q$) indicates a relatively high price (or workload size). A data centre operator has an incentive to run jobs when energy is cheap and is forced to run them when the enqueued workload is high. For this reason, the algorithm executes jobs if $index_Q \geq index_P$ (line 6). Figure 3 shows a graphical representation of the idea.

Jobs are also executed if the global state is considered favourable (line 6). The global state is assessed by the function `checkGlobalStatus`(...) which is given in Algorithm 2. A sliding window approach, as used for the local view,

might not be suitable during steady weather conditions and cannot take seasonal effects into account. The global view is therefore designed to widen the window without storing all the values. Instead of lists, the algorithm keeps a hash map A having one entry for each possible price (rounded to one decimal place). When a new price is noted, the respective value is incremented by one, and all entries are multiplied by an ageing factor f (line 1 - 3). For the evaluation, we have chosen f such that it halves the impact of a single event every six months. The variable $\texttt{sum}_{<p_c}$ ($\texttt{sum}_{>p_c}$) reflects the number of times a smaller (larger) energy price than p_c was observed. By defining an upper threshold on the workload size (line 7), the algorithm determines the current `fillRate` (line 8). If the squared `fillRate` is greater than the `priceRate` (line 6), `checkGlobalStatus`(...) returns `true`, otherwise `false` (line 9).

The decision is made by comparing the `fillRate` of the queue with the `priceRate` which is essentially the percentage of prices that are smaller than the current one. The `fillRate` is squared to reduce the influence of the global view and to strengthen the local view which should be the main criterion.

Algorithm 2 `checkGlobalStatus`(remAllocationtime, p_c)

1: **for all** p **do**
2: $\quad A[p] \leftarrow A[p] \cdot f$
3: $A[p_c] \leftarrow A[p_c] + 1$
4: $\texttt{sum}_{<p_c} \leftarrow \sum_{p<p_c} A[p]$
5: $\texttt{sum}_{>p_c} \leftarrow \sum_{p>p_c} A[p]$
6: $\texttt{priceRate} \leftarrow \texttt{sum}_{<p_c} \,/\, (\texttt{sum}_{<p_c} + \texttt{sum}_{>p_c})$
7: $\texttt{workloadLimit} \leftarrow 48 \cdot \#\texttt{cores}$
8: $\texttt{fillRate} \leftarrow \texttt{remAllocationtime} \,/\, \texttt{workloadLimit}$
9: **return** $\texttt{fillRate}^2 > \texttt{priceRate}$

The runtime complexity of Algorithm 1 is dominated by either the number of enqueued jobs or the size of L_P and L_Q. Summing up the remaining allocation time in line 1 has a runtime complexity linear in the number of jobs enqueued in all five queues. L_P and L_Q are implemented as double-linked lists. The updates

in line 2 and 3 are fast, but sorting the list has $O(n \cdot \log(n))$ runtime complexity (line 4 and 5) where the size n of the lists is defined by the sliding window. Let k be the number of scheduled jobs per scheduling step and c the number of nodes in the cluster. Then the for-loop of Algorithm 1 has a complexity of $O(k \cdot c)$.

The runtime complexity of Algorithm 2 is linearly bound by the size of the hash map A. Rounding the energy prices to one decimal place keeps the size of A small.

2.2.4. Green Scheduler

As described in Section 2.1, the green scheduler uses weather / energy forecasts to decide when the jobs are to be run. The weather forecasts, and subsequently also the energy forecasts, are provided for the next 48 hours and divided into one hour slots. The goal of the green scheduler is to maximise the utilisation of the most energy-efficient slots. Its workflow is described in Algorithm 3.

Similar to the global view of the MATH scheduler (Algorithm 2), the green scheduler calculates the fill rate of the system (line 1 - 3). In contrast to the constant value of 48 hours in Algorithm 2, `queueThreshold` is configurable. During the evaluation we use thresholds of 12 to 192 hours. The algorithm favours nearby time slots over distant ones by adding increasing penalties to the forecast values (line 6). The reasons are the improvement of the turnaround time and the inaccuracy of long-distance forecasts. It is usually not worth to wait longer for a small and uncertain benefit. Having access to a Smart Meter, the algorithm can correct the energy forecast for the next slot. Instead of relying on the previously generated energy forecast for the next 60 minutes, we replace it by the most recent Smart Meter measurement (line 7). This reduces the damage induced by forecasts of bad quality.

For each job `t` marks the slot at which the job's maximum waiting time is reached (line 10). $F_{\leq t}$ defines the subset of the energy forecast `F` containing all slots until `t` (line 11). These are the slots for which processing the job is optional. By sorting the energy values in $F_{\leq t}$ in ascending order and computing

Algorithm 3 Specific code of green scheduler

1: `queueLimit` $\leftarrow$ `queueThreshold` $\cdot$ #`cores`
2: `remAllocationtime` $\leftarrow \sum_{q \in queues} \sum_{j \in q} remainingAllocationtime(j)$
3: `fillRate` $\leftarrow$ `remAllocationtime` / `queueLimit`
4: $F \leftarrow$ energy forecast as hash map (hours to timeslot $\rightarrow$ energy value)
5: **for all** (`time`, `value`) $\in F$ **do**
6: $\quad F$[`time`] $\leftarrow$ `value` + `time` $\cdot$ 0.05
7: $F[0] \leftarrow$ current energy value
8: **for all** $job \in queues$ **do**
9: $\quad$ // round-robin queueing
10: $\quad t \leftarrow$ job's time to end of maximum waiting time
11: $\quad F_{\leq t} \leftarrow F$ limited by t
12: $\quad$ `index` $\leftarrow$ index of current value in $F_{\leq t}$ (sorted in ascending order)
13: $\quad$ `percBetterValues` $\leftarrow$ `index` / $|F_{\leq t}|$
14: $\quad$ **if** `percBetterValues` $\leq$ `fillRate` **then**
15: $\quad\quad$ **if** enough resources available **then**
16: $\quad\quad\quad$ dequeue job and process it

the index of the current energy value (line 12), the algorithm determines the percentage of slots that are better than the current one in $F_{\leq t}$ (line 13). The job is executed if the percentage is smaller than the fill rate and if sufficient resources are available (line 14-16). The idea is that also less favourable slots are utilised when the fill rate grows.

Due to the new multi-queue approach and preprocessing and postprocessing phases, the previous version of the green scheduler presented in [1] needed to be redesigned. Furthermore, the new maximum waiting time requires that the energy situation is individually assessed for every job.

The runtime complexity of the green scheduler is dominated by line 2 and by the for-loop at line 8. As discussed in Section 2.2.3, summing up the workload is linearly bound by the number of enqueued jobs. The second for-loop has a runtime complexity of $O(k \cdot (c + |F|))$, where k is the number of scheduled jobs

and c the number of nodes in the cluster. Calculating and updating the energy forecast can be done in a separate process and should not be part of the critical path (see Section 2.1.1). The time complexity of line 4 and of the first loop is therefore $O(|F|)$.

2.2.5. Enhanced Green Scheduler (ENHG)

As described in Section 2.1, the energy forecast turned out to be fairly inaccurate, which limits the efficiency of the green scheduler. For this reason, it is interesting to determine how well the green scheduler would work if it had access to a perfect energy forecast. We call this scheduler the *enhanced green scheduler*. Since the energy forecast is correct, the *penalty* is not applied.

2.2.6. Cost Optimal Scheduler (OPT)

While the enhanced green scheduler uses perfect energy forecasts, it is not optimal because it is limited by the 48 hour forecast window. The *cost optimal scheduler* has perfect knowledge of the future energy prices and uses it to create an (assumably) nearly optimal schedule. It can only be an approximation because the problem of scheduling jobs of different runtimes and resource requirements is known to be NP-hard [27].

The scheduling algorithm creates one hundred schedules and picks the best of them. For each of these schedules it generates a random list of all jobs and schedules one job after the other. For each job, the algorithm finds the best valid schedule while observing the maximum waiting time; i.e., the placement of the job is optimal conditioned on the placement of the previous jobs of the list.

This procedure is unlikely to produce an optimal schedule, but, due to the low utilisation of the clusters in our experiments, we expect a fairly small number of collisions and the approximation to be acceptable.

3. Evaluation

For the evaluation, we compare the scheduling strategies described in the previous section. They are executed in a simulated environment using real

Table 3: Percentiles of the energy price in ct/kWh

P(05)	P(10)	P(25)	P(50)	P(75)	P(90)	mean
10.0	11.35	14.8	17.25	18.9	20.0	16.49

weather and energy data and workload traces. To quantify the quality of the strategies, we use three measures: the first is the share of green energy, the second is the energy costs in our hypothetical price model and the third is the average turnaround time of the jobs. Before summarising the execution and the outcomes of the experiments, we start by describing and motivating the price model.

3.1. Price Model

As pointed out before, the grid provider has an incentive to reduce the local surplus and sell the energy where it is produced. For this reason, we assume that the customer price will drop once the locally generated energy exceeds the customer demand and model the price as a function $p(x)$ of the grid exchange x between the Paderborn grid and the HV grid (see Section 2.1.4):

$$p(x) = R + y(x).$$

R is the reference price of 15 ct/kWh and $y(x)$ the dynamic portion defined as follows:

$$y(x) = \begin{cases} -V, & \text{if } x \leq -10 \\ V, & \text{if } x \geq 10 \\ \frac{x}{10} \cdot V, & \text{otherwise} \end{cases}$$

where the variance V is set to $V = \frac{1}{3} \cdot R = 5$ ct/kWh.

According to *Eurostat*[16], the 15 ct/kWh baseline is the average price for small industry consumers with an annual energy consumption of 20 to 500 MWh in Germany. This price includes all non-recoverable taxes and levies.

[16] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nrg_pc_205&lang=en

Since the prices of the EPEX[17] intra day spot market usually vary by more than 50 €/MWh, we consider the variance of 33 % or 5 ct/kWh a conservative estimate. Increasing or decreasing the variance would have an impact on the achievable savings, but, regardless of the exact value, the cost analysis would show the same trend. Table 3 displays the percentiles of the resulting energy price for the simulation period.

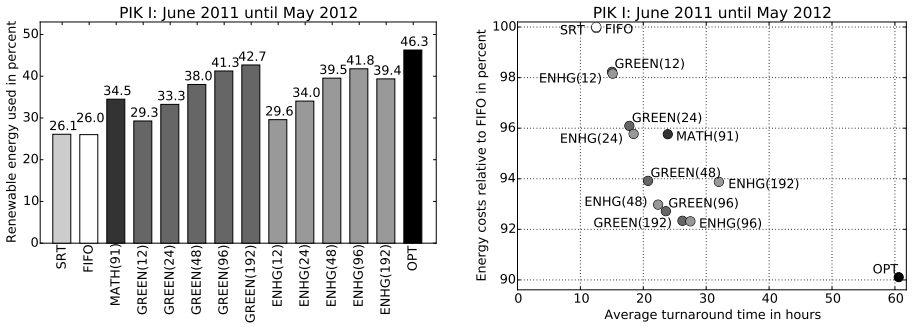
3.2. Simulation

To simulate a realistic data centre, we take workload logs from the *Potsdam Institute for Climate Impact Research*[18] (PIK). These are the same traces that we used in [1]. In addition, we run simulations on the traces from the *Narwhal* cluster which was part of the *Sharcnet*[19][20]. Each simulation covers a full year and is executed in one hour steps. We assume that the batch jobs executed are CPU-intensive and consume the full *thermal design power* (TDP) of the assigned cores.

The scheduling strategies of Section 2.2 are simulated with different configurations. The MATH scheduler is run with sliding window sizes of 7, 28, 91 and 182 days and the (enhanced) green scheduler with queue sizes of 12, 24, 48, 96 and 192 hours. While the differences among the MATH scheduler results were already small in our previous paper [1] (with the exception of the 7 days window), they are now, after improving the scheduler and introducing the multi-queue model, so insignificant that we will only discuss the 91 days configuration as a representative of the MATH scheduler.

Since these schedulers achieve improvements by delaying the processing of jobs during unfavourable time periods, unfinished jobs remain in the queues at the end of the simulation. The differences in the computation time of these schedulers are evened out by charging 20 ct/kWh for the additional computation

[17] http://www.epexspot.com
[18] http://www.cs.huji.ac.il/labs/parallel/workload/l_pik_iplex
[19] https://www.sharcnet.ca
[20] http://www.cs.huji.ac.il/labs/parallel/workload/l_sharcnet/index.html



(a) Share of renewable energy (PIK I) (b) Costs versus turnaround time (PIK I)

Figure 4: Comparison of the PIK I simulation results with different configurations

time.

3.2.1. PIK Traces

The *PIK cluster* is a *320 node IBM iDataPlex Cluster* and has 2560 cores. The *Intel Xeon Harpertown CPU* has a TDP between 50 and 150 watts. We simulated the cluster assuming a TDP of 120 watts per CPU and 30 watts per core. Based on measurements in our own data centre, we set a server's idle power consumption to 80 watts and the power-off / *Wake on LAN* consumption to 7 watts. The workload logs contain 742965 jobs using up to 1024 cores and having individual runtimes between a few seconds and 30 days. We extracted three traces that we name *PIK I*, *PIK II* and *PIK III*. They cover the period between the 1st June and the 31st May of the years 11/12, 10/11 and 09/10, respectively.

Table 4, 5 and 6 show the results of the *PIK* simulations for the proposed scheduling strategies. The first column of each table names the scheduling strategy. Column 2 and 3 show the total energy costs and these costs normalised with respect to the FIFO scheduler. Hence, it represents the cost reduction of the respective scheduling strategy. The average turnaround time is displayed in the forth column and the percentage of renewable energy in the fifth column.

Figures 4, 5 and 6 are graphical representation of the results. The bar charts

PIK I - 11/12				
Strategy	Costs in EUR	Cost ratio	TAT in hours	Renewable energy share
SRT	51835	1.000	12.53	26.10
FIFO	51855	1.000	12.52	26.01
MATH(91)	49658	0.957	23.89	34.50
GREEN(12)	50937	0.982	15.03	29.28
GREEN(24)	49829	0.961	17.78	33.26
GREEN(48)	48701	0.939	20.75	38.03
GREEN(96)	48079	0.927	23.59	41.26
GREEN(192)	47882	0.923	26.22	42.70
ENHG(12)	50899	0.982	15.13	29.60
ENHG(24)	49663	0.958	18.46	34.05
ENHG(48)	48214	0.930	22.35	39.54
ENHG(96)	47871	0.923	27.49	41.79
ENHG(192)	48681	0.939	32.01	39.37
OPT	46726	0.901	60.60	46.28

Table 4: PIK I - 11/12 simulation results using different scheduling strategies and configurations

display the share of renewable energy consumption achieved by each scheduler and the scatter plot sets the cost efficiency in relation to the turnaround time.

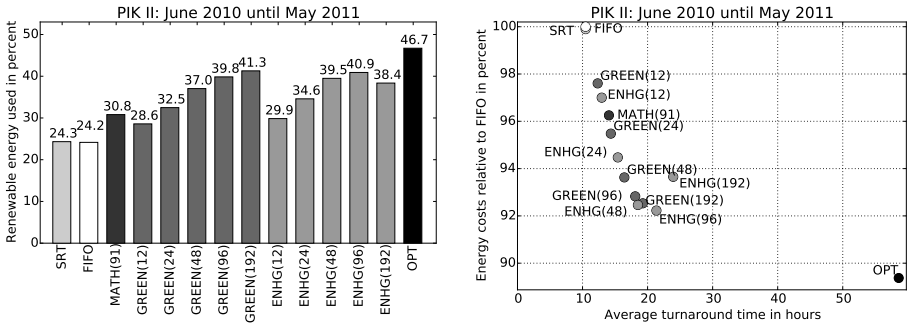
During the PIK I simulation, the FIFO scheduler needs 7.058 million core hours and energy for € 51855. The average turnaround time of the finished jobs is 12.52 hours, and 26.01 % of the consumed energy is locally generated. Almost identical results are accomplished with the shortest remaining time scheduler. For the MATH scheduler, we observe that the use of green energy increases to 34.50 % and that the costs are reduced by 4.3 %. This comes at the costs of an increased average turnaround time of eight hours. The green scheduler can improve the usage of renewable energy and cost savings further. For the longest queue size of 192 hours, the share of renewable energy can be increased to

PIK II - 10/11				
Strategy	Costs in EUR	Cost ratio	TAT in hours	Renewable energy share
SRT	39210	0.999	10.47	24.32
FIFO	39250	1.000	10.47	24.15
MATH(91)	37779	0.963	14.06	30.81
GREEN(12)	38309	0.976	12.33	28.58
GREEN(24)	37475	0.955	14.35	32.48
GREEN(48)	36749	0.936	16.43	37.04
GREEN(96)	36434	0.928	18.10	39.84
GREEN(192)	36319	0.925	19.26	41.29
ENHG(12)	38072	0.970	12.95	29.85
ENHG(24)	37080	0.945	15.43	34.58
ENHG(48)	36291	0.925	18.51	39.50
ENHG(96)	36199	0.922	21.36	40.90
ENHG(192)	36758	0.937	23.94	38.36
OPT	35079	0.894	58.60	46.71

Table 5: PIK II - 10/11 simulation results using different scheduling strategies and configurations

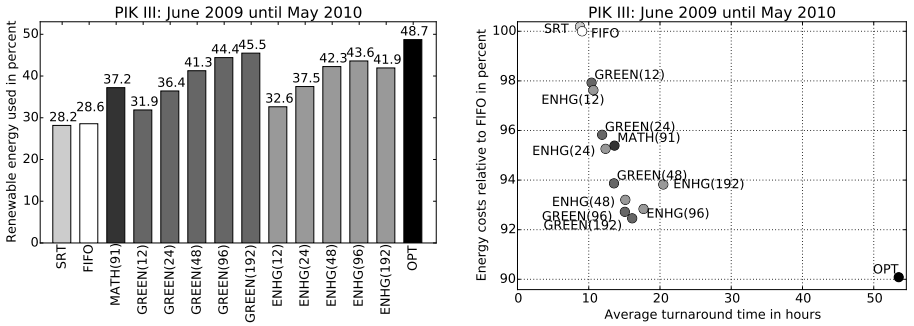
42.70 % which results in cost savings of 7.7 %. Comparing the MATH scheduler with the 96 hour green scheduler, which have a similar turnaround time, shows the benefit of the energy forecast. The green scheduler achieves 3.1 % higher cost savings and 8.2 % higher utilisation of renewable energy at a slightly lower turnaround time.

Interestingly, the enhanced green scheduler performs best with a 96 hour queue. Compared to ENHG(96), ENHG(192) has a lower cost efficiency and a lower share of renewable energy consumed. The reason is that, if the queue is relatively full, many jobs will pass the maximum waiting time. (For a full 192 hours queue, the cluster requires 192 hours to process all jobs, and the maximum waiting time is only 96 hours.) This limits the scheduler's ability to shift the



(a) Share of renewable energy (PIK II) (b) Costs versus turnaround time (PIK II)

Figure 5: Comparison of the PIK II simulation results with different configurations



(a) Share of renewable energy (PIK III) (b) Costs versus turnaround time (PIK III)

Figure 6: Comparison of the PIK III simulation results with different configurations

workload and reduces its efficiency.

As a penalty is applied to the energy forecast of the green scheduler (but not the enhanced green scheduler), the green scheduler shows a better average turnaround time in general and the impact of the above-mentioned effect is smaller for the GREEN(192) scheduler.

The cost-optimal scheduler achieves maximum cost savings of 9.9 % which is only 2.2 % more than the best green scheduler results. With a queue size of 48 to 192 hours, the green scheduler performs well.

The PIK II and III simulation results show the same trend, as one can see in

PIK III - 09/10				
Strategy	Costs in EUR	Cost ratio	TAT in hours	Renewable energy share
SRT	45632	1.002	8.77	28.16
FIFO	45548	1.000	9.06	28.57
MATH(91)	43448	0.954	13.62	37.21
GREEN(12)	44604	0.979	10.40	31.87
GREEN(24)	43647	0.958	11.88	36.42
GREEN(48)	42755	0.939	13.54	41.27
GREEN(96)	42232	0.927	15.07	44.41
GREEN(192)	42113	0.925	16.10	45.48
ENHG(12)	44465	0.976	10.65	32.63
ENHG(24)	43389	0.953	12.36	37.48
ENHG(48)	42452	0.932	15.13	42.26
ENHG(96)	42285	0.928	17.69	43.61
ENHG(192)	42734	0.938	20.46	41.94
OPT	41032	0.901	53.50	48.72

Table 6: PIK III - 09/10 simulation results using different scheduling strategies and configurations

Figure 4b, 5b and 6b. The MATH scheduler, green scheduler and enhanced green scheduler find a good trade-off between green energy utilisation and turnaround time.

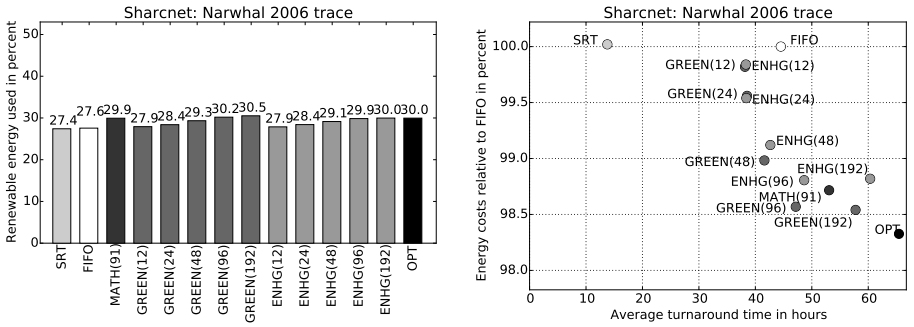
Table 7 shows a comparison of the results of the improved scheduler with the previously published ones [1]. The surprising increase in the turnaround time of the FIFO scheduler can be explained by the job distribution in the newly introduced multi-queue model. The 292184 jobs of the PIK I trace are divided into the queues as follows: tiny: 184256 (63 %); short: 41743 (14 %); short-node: 6254 (2 %), long: 57178 (20 %), long-node: 2753 (1 %). The different queue lenghths are ignored when the jobs are taken out of the them in a round-robin

Comparison of old and updated strategies				
	Old Strategy Results		New Strategy Results	
Strategy	Cost ratio	TAT in hours	Cost ratio	TAT in hours
FIFO	1.000	11.58	1.000	12.52
MATH(91)	0.936	32.10	0.958	23.89
GREEN(3)	0.914	34.11	-	-
GREEN(12)	0.900	45.20	0.982	15.03
GREEN(24)	0.884	62.69	0.961	17.78
GREEN(48)	0.863	123.36	0.939	20.75
GREEN(96)	-	-	0.927	23.59
GREEN(192)	-	-	0.923	26.22
ENHG(3)	0.918	28.82	-	-
ENHG(12)	0.901	32.46	0.982	15.13
ENHG(24)	0.882	60.85	0.958	18.46
ENHG(48)	0.863	138.71	0.930	22.35
ENHG(96)	-	-	0.923	27.49
ENHG(192)	-	-	0.939	32.01
OPT	0.831	389.39	0.901	60.60

Table 7: Comparison of the cost efficiency and turnaround time of the old and new strategies for the PIK 11/12 trace

fashion. This is not a problem in off-peak situations when all the queues are processed completely regardless of their initial size, but peak hours seem to be sufficient to considerably increase the average turnaround time.

Compared to the results of [1], cost efficiency and turnaround time of all green schedulers are lower. It seems as if the newly introduced maximum waiting time has shrunken the spectrum of possible results. However, taking the turnaround time into account, the new results can be regarded as better. The best previous turnaround time was achieved by the GREEN(3) scheduler: TAT 2.9 times



(a) Share of renewable energy (b) Costs versus turnaround time

Figure 7: Comparison of the Narwhal simulation results with different configurations

higher compared to FIFO, 8.6 % cost efficiency. In comparison, in the new system the worst turnaround time was reached by the GREEN(192) scheduler which has a turnaround time factor of roughly 2.1 and achieves cost savings of 7.7 %.

Even more important than the improved TAT-cost trade-off is the achieved flexibility. By increasing the maximum waiting time the new version could achieve the same cost savings, but the old version is not able to achieve comparable turnaround times. For systems with hard deadlines or complex priority schemes, it is now feasible to define a maximum waiting time for each job to have a fast response time where necessary, while being cost-efficient where possible.

3.2.2. Narwhal Trace

The *Narwhal* cluster[21] consists of 267 nodes each equipped with two dual-core Opteron processors. The simulated power consumption of every cores is 45 watts; the idle power consumption is 80 watts per node. This results in a maximum power consumption of 260 watts per node. The simulation executes the trace from 2006 which consists of 266785 jobs. The runtimes of the jobs range from a few seconds to a few days, and the majority of jobs allocate a single core only.

[21] `https://www.sharcnet.ca/help/index.php/Narwhal`

Narwhal				
Strategy	Costs in EUR	Cost ratio	TAT in hours	Renewable energy share
SRT	71248	1.000	13.78	27.41
FIFO	71234	1.000	44.52	27.57
MATH(91)	70319	0.987	53.07	29.94
GREEN(12)	71105	0.998	38.16	27.92
GREEN(24)	70920	0.996	38.53	28.39
GREEN(48)	70509	0.990	41.60	29.34
GREEN(96)	70214	0.986	47.16	30.19
GREEN(192)	70194	0.985	57.75	30.52
ENHG(12)	71120	0.998	38.32	27.87
ENHG(24)	70906	0.995	38.39	28.41
ENHG(48)	70607	0.991	42.65	29.15
ENHG(96)	70383	0.988	48.65	29.85
ENHG(192)	70392	0.988	60.34	29.98
OPT	70042	0.983	65.44	29.96

Table 8: Narwhal simulation results using different scheduling strategies and configurations

Table 8 and Figure 7 present the simulation results. While the absolute numbers are disappointing regarding green energy utilisation, the strategies show the same trend that has been observed for the Potsdam traces. In our simulation the PIK traces result in an average cluster utilisation of 23 % to 32 %, while the Narwhal trace implies a utilisation of 68 %. The jobs enter the system in bursts and do not permit any relevant workload shifting within the allowed waiting time. As a result, the optimal scheduler achieves costs savings of only 1.7 %.

210887 out of the 266785 jobs are regarded “tiny”, for which reason the shortest remaining time scheduler has a much better turnaround time than the FIFO scheduler. Even some of the (enhanced) green schedulers have a better average turnaround time than the FIFO scheduler. This can be explained by

PIK I - 11/12				
Strategy	Preemptive cost ratio	Non-preemptive cost ratio	Preemptive TAT in h	Non-preemptive TAT in h
SRT/SJF	1.000	1.000	12.53	12.13
FIFO	1.000	1.000	12.52	12.22
MATH(91)	0.957	0.997	23.89	12.41
GREEN(12)	0.982	0.985	15.03	14.25
GREEN(24)	0.961	0.978	17.78	15.11
GREEN(48)	0.939	0.973	20.75	15.92
GREEN(96)	0.927	0.968	23.59	16.92
GREEN(192)	0.923	0.965	26.22	17.76

Table 9: Simulation results using nonpreemptive job scheduling

the postprocessing phase of the green scheduler, which processes additonal jobs of the tiny queue.

All in all, there is only little room for improvement when a large share of workloads cannot be shifted. Yet, it is interesting to see how the algorithms behave under circumstances very different from the PIK traces and that they still perform well.

3.3. Non-preemptive Simulation

In the previous sections we assumed that jobs can be interrupted at any time. We have rerun the PIK I simulation under the assumption that the jobs are non-preemptive.

Table 9 lists the results of the MATH scheduler, the green scheduler and the FIFO scheduler. It compares the cost ratio and turnaround time of the preemptive case from Table 4 with the values of the non-preemptive case. While the values of the MATH scheduler nearly degrade to the level of the FIFO scheduler for non-preemptive jobs, the green scheduler is still able to reduce the costs by up to 3.5 % compared to the FIFO results.

3.4. Summary

Considering the large average error of the energy forecast, the green schedulers' results are surprisingly close to the enhanced green scheduler results. This indicates that a reasonable estimate of the expected energy situation is enough to make an informed decision, as long as a Smart Meter can be queried to correct forecast errors at runtime. This is an important outcome and allows the conclusion that even a simple forecast with an error margin of up to 30 % can yield good results. Whether a better forecast is worth the higher operational overhead and potentially expensive input data depends on the size of the data centre and, neglecting the environmental aspect, the possible cost savings. In cases where the energy price is difficult to predict, the MATH scheduler might still be able to improve the data centre efficiency.

As a matter of principle, the strategies need to be able to postpone job processing. In Section 3.2.2 and 3.3 we evaluated the effects of high cluster utilisation and non-preemptible jobs and showed that the efficiency gains are much lower. However, these unfavourable scenarios show that the proposed strategies still do better compared to FIFO scheduling.

4. Conclusion

This paper has presented and evaluated two Smart Grid-aware scheduling strategies. We assessed the increase in the utilisation of locally produced renewable energy and the monetary benefit based on a hypothetical, yet conservatively estimated price model. We also quantified the performance penalty in terms of a higher turnaround time. The assumed Smart Grid is based on real measurements of Paderborn's energy grid, and, to the best of our knowledge, this paper is the first in making predictions for a Smart Grid itself while most of the previous work predicted the generation of on-site power plants using the grid merely as a backup energy source.

For the more simple scheduler based on a mathematical model, recording Smart Meter values is already sufficient to increase the share of consumed

renewable energy to 34.5 % compared to 26.0 % of the reference FIFO scheduler running the PIK 11/12 trace. The *green scheduler* also requires weather data to predict the future energy surplus in the grid and to schedule the incoming jobs accordingly. While it is possible to further improve the use of renewable energy to 42.7 %, the downside is a rise of the average turnaround time by a factor of up to 2.1. Our approximation of a cost optimal scheduler achieves a renewable energy consumption of 46.3 %, at the costs of an increased average turnaround time of 4.8 %.

Compared to the prior versions of the strategies presented in [1], the new MATH scheduler and green scheduler show an improved trade-off between green energy utilisation and turnaround time. We added two parameters, the maximum waiting time of the jobs and the forecast penalty, which can be used to tune the algorithms. The maximum waiting time determines for how long a job can be postponed. If it is too short, it limits the energy-optimised placement of the jobs; if it is too long, the throughput might be too low. The forecast penalty downgrades predicted energy values subject to their forecast time. A slightly worse energy current value can thus be preferred to a value in the (distant) future.

Our future work will focus on the influence of hard deadlines and job priorities on the scheduling strategies. Furthermore, we will refine our strategies, once we can test them in a real Smart Grid environment.

5. Acknowledgment

This work was supported by the German Ministry of Economic Affairs and Energy (BMWi) under Grant 01ME11067 (project “GreenPAD”).

We would like to thank *Weser Westfalen Energy* (WWE) and especially Uwe Granzow for providing and explaining the energy data to us. The WWE was one of the partners in the BMWi project “GreenPAD”.

References

[1] M. Mäsker, L. Nagel, A. Brinkmann, F. Lotfifar, M. Johnson, Smart grid-aware scheduling in data centres, in: 2015 Sustainable Internet and ICT for Sustainability, SustainIT 2015, Madrid, Spain, April 14-15, 2015, IEEE, 2015, pp. 1–9. doi:10.1109/SustainIT.2015.7101362.
URL http://dx.doi.org/10.1109/SustainIT.2015.7101362

[2] No author given, Die Energie der Zukunft, Tech. rep., Bundesministerium für Wirtschaft und Energie (BMWi) (December 2014).

[3] C. Li, W. Zhang, C.-B. Cho, T. Li, Solarcore: Solar energy driven multi-core architecture power management, in: High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on, IEEE, 2011, pp. 205–216.

[4] C. Li, A. Qouneh, T. Li, iswitch: coordinating and optimizing renewable energy powered server clusters, in: Computer Architecture (ISCA), 2012 39th Annual International Symposium on, IEEE, 2012, pp. 512–523.

[5] Í. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, R. Bianchini, Greenslot: scheduling energy consumption in green datacenters, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2011, p. 20.

[6] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Green-hadoop: leveraging green energy in data-processing frameworks, in: Proceedings of the 7th ACM european conference on Computer Systems, ACM, 2012, pp. 57–70.

[7] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Generation Computer Systems 28 (5) (2012) 755–768. doi:10.1016/

j.future.2011.04.017.
URL http://dx.doi.org/10.1016/j.future.2011.04.017

[8] W. Deng, F. Liu, H. Jin, B. Li, D. Li, Harnessing renewable energy in cloud datacenters: opportunities and challenges, IEEE Network (2014) 48–55.

[9] M. Lange, Analysis of the uncertainty of wind power predictions, Ph.D. thesis, Universität Oldenburg (2003).

[10] B. G. Brown, R. W. Katz, A. H. Murphy, Time series models to simulate and forecast wind speed and wind power, Journal of Climate and Applied Metereology 23.

[11] R. Baïle, J. F. Muzy, P. Poggi, Short-term forecasting of surface layer wind speed using a continuous random cascade model, Wind Energy 14 (6) (2011) 719–734. doi:10.1002/we.452.
URL http://dx.doi.org/10.1002/we.452

[12] M. Mohandes, T. Halawani, S. Rehman, A. A. Hussain, Support vector machines for wind speed prediction, Renewable Energy 29 (6) (2004) 939–947.

[13] A. Kusiak, H. Zheng, Z. Song, Short-term prediction of wind farm power: a data mining approach, Energy Conversion, IEEE Transactions on 24 (1) (2009) 125–136.

[14] F. Cassola, M. Burlando, Wind speed and wind energy forecast through kalman filtering of numerical weather prediction model output, Applied Energy 99 (2012) 154–166.

[15] G. Giebel, R. Brownsword, G. Kariniotakis, M. Denhard, C. Draxl, The State-Of-The-Art in Short-Term Prediction of Wind Power: A Literature Overview, 2nd edition, ANEMOS.plus, 2011, project funded by the European Commission under the 6th Framework Program, Priority 6.1: Sustainable Energy Systems.

[16] C. Wang, N. Nasiriani, G. Kesidis, B. Urgaonkar, Q. Wang, L. Y. Chen, A. Gupta, R. Birke, Recouping energy costs from cloud tenants: Tenant demand response aware pricing design, in: Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems, e-Energy '15, ACM, New York, NY, USA, 2015, pp. 141–150. doi:10.1145/2768510.2768541.
URL http://doi.acm.org/10.1145/2768510.2768541

[17] R. S. Brewer, N. Verdezoto, M. K. Rasmussen, J. M. Entwistle, K. Grønbæk, H. Blunck, T. Holst, Challenge: Getting residential users to shift their electricity usage patterns, in: Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems, e-Energy '15, ACM, New York, NY, USA, 2015, pp. 83–88. doi:10.1145/2768510.2770934.
URL http://doi.acm.org/10.1145/2768510.2770934

[18] M. Brown, J. Renau, Rerack: Power simulation for data centers with renewable energy generation, ACM SIGMETRICS Performance Evaluation Review 39 (3) (2011) 77–81.

[19] C. Ren, D. Wang, B. Urgaonkar, A. Sivasubramaniam, Carbon-aware energy capacity planning for datacenters, in: Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 391–400. doi:10.1109/MASCOTS.2012.51.
URL http://dx.doi.org/10.1109/MASCOTS.2012.51

[20] N. Sharma, S. Barker, D. Irwin, P. Shenoy, Blink: managing server clusters on intermittent power, in: ACM SIGPLAN Notices, Vol. 47, ACM, 2011, pp. 185–198.

[21] B. Aksanli, J. Venkatesh, L. Zhang, T. Rosing, Utilizing green energy prediction to schedule mixed batch and service jobs in data centers, ACM SIGOPS Operating Systems Review 45 (3) (2012) 53–57.

[22] Y. Zhang, Y. Wang, X. Wang, Greenware: Greening cloud-scale data centers to maximize the use of renewable energy, in: F. Kon, A.-M. Kermarrec (Eds.), Middleware 2011, Vol. 7049 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 143–164. doi:10.1007/978-3-642-25821-3_8.
URL http://dx.doi.org/10.1007/978-3-642-25821-3_8

[23] S. Akoush, R. Sohan, A. Rice, A. W. Moore, A. Hopper, Free lunch: exploiting renewable energy for computing, in: HotOS'13 Proceedings of the 13th USENIX conference on Hot topics in Operating Systems, 2011, pp. 17–17.

[24] J. L. Berral, Í. Goiri, T. D. Nguyen, R. Gavalda, J. Torres, R. Bianchini, Building green cloud services at low cost, in: IEEE 34th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2014, pp. 449–460.

[25] Y. Li, D. Chiu, C. Liu, L. T. Phan, T. Gill, S. Aggarwal, Z. Zhang, B. T. Loo, D. Maier, B. McManus, Towards dynamic pricing-based collaborative optimizations for green data centers, in: Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on, IEEE, 2013, pp. 272–278.

[26] O. Niehörster, A. Keller, A. Brinkmann, An energy-aware saas stack, in: IEEE 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011, pp. 450–453. doi:10.1109/MASCOTS.2011.52.

[27] J. Lenstra, D. Shmoys, E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Mathematical Programming 46 (1-3) (1990) 259–271. doi:10.1007/BF01585745.
URL http://dx.doi.org/10.1007/BF01585745

[28] L. Schrage, A proof of the optimality of the shortest remaining processing time discipline, Operations Research 16 (1968) 687–690. doi:10.1287/

opre.16.3.687.

URL http://ci.nii.ac.jp/naid/30041575850/en/