Master's Thesis

Renewable-Aware Kubernetes Scheduling to Mitigate Carbon Emissions in Distributed Systems

Timo Kraus
Matriculation Number 399111

Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Distributed and Operating Systems

Reviewers:
Prof. Dr. habil. Odej Kao
Prof. Dr. Volker Markl

Supervised by:
M. Sc. Philipp Wiesner

Berlin, 06.01.2022

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used. The independent and unaided completion of the thesis is affirmed by affidavit.

Berlin, 06.01.2022 Timo Kraus

Abstract

Um den Klimawandel einzudämmen und das vom IPCC berechnete 1.5°C Ziel einzuhalten sind erhebliche CO_2 Reduktionen in allen Bereichen der Industrie und Gesellschaft notwendig. Diesbezügliche Bestreben im IT Sektor zeigen, dass große Cloud Computing Anbieter hierfür bereits auf den Einsatz erneuerbarer Energien setzen und ihre Rechenzentren mit entsprechenden lokalen Erzeugern wie beispielsweise Photovoltaikanlagen ausstatten. Da Rechenzentren, vor allem auch durch das aufstrebende Fog Computing Paradigma, meist geographisch verteilt sind, kann die Verfügbarkeit erneuerbarer Energie bei einer heterogenen Austattung mit Erzeugern über die verschiedenen Standorte hinweg deutliche Unterschiede aufweisen. Da moderne Software Plattform-agnostisch durch Containervirtualisierung betrieben wird, hat sich hierfür das Container Management System Kubernetes in den letzten Jahren zum de-facto Standart im Cloud Computing Bereich entwickelt. Bei bestehenden Kubernetes Scheduler Lösungen werden lokal verfügbare erneuerbare Energien und entpreschende Vorhersagen allerdings bei der Auswahl geeigneter Nodes nicht ausreichend berücksichtigt.

Energieerzeugungs- und Verbrauchsdaten, sowie Vorhersagen an lokalen Rechenzentren sollen deshalb durch eine Erweiterung des Kubernetes Scheduling Algorithmus in Form des hier entwickelten green-k8s-scheduler integriert werden, um den Standort mit der höchsten Verfügbarkeit an erneuerbarer Energie zu begünstigen. Des Weiteren ist der green-k8s-descheduler in der Lage auf wechselnde Wetterbedingungen zu reagieren und somit auch langfristig die Platzierung der Anwendungen an den grünsten Standorten zu gewährleisten. Beide Komponenten wurden durch multiple Tests auf einem Kubernetes Cluster mit geographisch verteilten Nodes evaluiert, wobei für jede Node die Verfügbarkeit von erneuerbarer Energie aus unterschiedlichen Quellen simuliert wurde. Die Ergebnisse zeigen, dass durch den Einsatz des green-k8s-schedulers, bzw. green-k8s-deschedulers eine Steigerung von bis zu 116% bei der Nutzung von erneuerbaren Energien im Vergleich zum Standard kube-scheduler erzielt werden konnte.

Abstract

To mitigate climate change and meet the 1.5°C target calculated by the IPCC, significant CO_2 reductions are needed in all domains. Efforts in the IT sector show that large cloud computing providers are already focusing on the use of renewable energies and equipping their data centers with corresponding local appliances such as photo-voltaic systems. Since data centers are usually geographically distributed, especially when considering the emerging fog computing paradigm, the availability of renewable energy can vary significantly across different locations. In order to use energy as efficiently as possible directly at the point of generation, this thesis presents a software-based approach to place workloads at the location where the greatest amount of green energy is expected to be available. As modern software is platform-agnostic through container virtualization, the container management system Kubernetes has become the de-facto standard in the field of cloud computing in recent years. However, existing Kubernetes schedulers lack integration of local renewable energy data and respective forecasts.

To favor the data center location with the highest availability of renewables, the green-k8s-scheduler is introduced. The extension of the Kubernetes scheduling algorithm takes energy production and consumption data, as well as predictions into account to favor the node in the cluster that is about to cover the workloads consumption with the highest possible share of renewable energy. Additionally, the green-k8s-descheduler is able to react to changing weather conditions and thus also ensures the optimal placement of applications in the long term. Both components were evaluated through several tests on a Kubernetes cluster with multiple geo-distributed nodes while simulating the availability of renewable energy from different sources for each node. The results show that by using the green-k8s-scheduler and green-k8s-descheduler, an increase of up to 116% in the use of renewable energy compared to the default kube-scheduler could be achieved.

Contents

1 Introduction

1.1 Motivation

Climate change is an omnipresent problem that requires worldwide action. As the earth's atmosphere continues to heat up, natural ecosystems and economies will be disrupted or destroyed in the near future. This temperature increase is a consequence of the greenhouse effect. The greenhouse effect is caused by greenhouse gases that are gathered in our atmosphere, as they are reflecting thermal radiation back to earth and lead to a rise in surface temperature. One of the main greenhouse gases is CO_2. It has a low decay rate, which means that it remains in the atmosphere for a very long time. The goal in current international agreements, e.g. the Paris Agreement, is the limitation of global warming to 1.5°C, compared to pre-industrial levels. According to a special report of the Intergovernmental Panel on Climate Change, the atmosphere can absorb, calculated from end-2017, no more than 420 Gt of CO_2 to reach the 1.5°C target [1]. However, since around 42 Gt of CO_2 is emitted globally every year, this budget is expected to be used up in less than seven years. Thus, society and industry need to strive towards a sustainable use of available resources to minimize the environmental footprint. One approach to lower carbon emissions is to maximize the use of renewable energy, as they have significantly lower lifecycle carbon emissions as traditional energy sources like coal or gas [2].

End consumers' awareness of the environment has increased significantly in recent years. At the same time, laws and regulations in this area have become more stringent. This increases the pressure on companies to make their business practices sustainable. In the field of cloud computing, the demand for data and digital services - and with it its associated carbon footprint - is expected to continue its exponential growth over the coming years. Even though virtualization technologies led to an increase in efficiency of data centers, cloud computing still accounted for about 1% of global electricity consumption in 2019 [3]. With the growth of the Internet of Things (connections are expected to double from 12 billion to 25 billion) and mobile internet users (projected to increase from 3.8 billion in 2019 to 5 billion by 2025), the trend will most likely lead to an increased use of distributed data centers [4]. As cloud native adoption becomes mainstream, the use of containers and orchestration engines like

Kubernetes grows accordingly [5]. Extending these existing tools with sustainability features could help to make cloud, fog or edge computing greener. By enforcing resource allocation based on the availability of renewable energy, the carbon intensity of the consumed energy of cloud services can be reduced. Considering economical (carbon pricing) and ecological (climate change) impacts of CO_2 emissions, the demand for solutions to mitigate carbon emissions will be of cross-industrial interest. Even though vendors of hyperscale data centers like Google[1] are already working towards operating on carbon free energy by purchasing renewable power from the grid, there is still a lack of available practical approaches for exploiting the location based availability of renewable energy in Kubernetes clusters. The share of electricity from weather-dependent wind and solar energy in the power mix keeps growing and the flexibility on the demand side needs to keep up. In today's cloud computing landscape, many applications are geo-distributed and containerized. With Kubernetes there is already a mature tool, to orchestrate these containers among clusters and distributed data centers on a massive scale. Kubernetes already has an internal scheduler implementation that automatically schedules so-called Pods, based on several factors like resource requirements or hardware, software or special policy constraints. However, the so-called kube-scheduler is considering mostly compute-related measures, but does not take external data, like the current renewable energy generation at the node location into account. To enable more sustainable scheduling decisions, this external data needs to be exposed to Kubernetes to be considered in the scheduling decision making process without violating quality of service requirements.

1.2 Research Objective

In this thesis, one goal is to point out the importance of tackling climate change and how state of the art technology can help to reduce carbon emissions by exploiting the local availability of renewable energy across geo-distributed Kubernetes clusters or node locations respectively. The core of this thesis is the implementation and evaluation of a kube-scheduler extension, which can take external renewable energy data into consideration during its scheduling process. Furthermore, an approach for adaptive rescheduling of pods will be tested in order to be able to react to changing weather conditions. To enable a well-informed decision making of the scheduler, it is further required to collect and prepare the data that is suitable to provide renewable energy generation data for the objective to maximize its overall usage. This data needs to be accessible by the kube-scheduler, which is then in charge of placing Pods on the greenest available node. For the implementation, only functionalities provided by Kubernetes

1 https://www.blog.google/inside-google/infrastructure/data-centers-work-harder-sun-shines-wind-blows/, *accessed 2021-09-17*

are to be used and extended in order to carry on the open-source idea of the project and to make adoption in productive use-cases as easy as possible. The following criteria are declared necessary for the achievement of the goal:

1. Generation of realistic real-time and forecast data of locally available renewable energy equipment like wind turbines or solar panels.
2. Provisioning of the aforementioned data in the Kubernetes cluster, or at the geo-distributed nodes respectively.
3. Review and selection of a suitable method to extend the kube-scheduler.
4. Implementation of a kube-scheduler extension, which considers renewable energy data in the scheduling process.
5. Implementation of a descheduler, which enables an adaptive rescheduling of Pods based on changing availability of renewable energy.
6. Analysis of resulting (de-)scheduling decisions in comparison to the default scheduler

While previous work has addressed the integration of renewable energy into distributed systems [6–19], few have implemented software-side integration through the use of existing container orchestration technologies such as Kubernetes [20, 21]. On top of that, the use of adaptive rescheduling and forecast data further differentiates this work from others. Through the interaction of the components mentioned above, the volatility of renewable energies is taken into account as efficiently as possible. The listed points are all evaluated on a test cluster that is set up with kOps on virtual machines in AWS (not as a managed Kubernetes service). This offers the possibility of arbitrary scaling and full access rights for configuring the Kubernetes control plane components.

1.3 Thesis Outline

This thesis is divided into six main chapters. After the introduction, chapter 2 discusses topics relevant to the thesis in order to enable a classification in the overall context. This includes the current state of climate research, an introduction to distributed systems, and a detailed description of Kubernetes as a technical foundation. Chapter 3 analyzes various literature from related topics for overlap with this thesis, as well as identifying a distinction from similar projects. Chapter 4 then deals with the design and implementation of the renewable energy

data simulator, the Kubernetes scheduler extension and the descheduler. Here, all system components and their functionality are described in detail. Chapter 5 then describes the test scenarios for the developed components and shows the behavior of the scheduler in use. Differences to the default scheduler are made clear and the advantages and improvements of the present solution are worked out. Finally, Chapter 6 discusses the results of the work critically and gives an outlook on possible further developments and application scenarios.

2 Fundamentals

2.1 Climate Change and the Impact of CO_2 Emissions

This chapter gives an overview of climate related topics in the context of this thesis. Chapter 2.1.1 provides a brief explanation of the greenhouse effect and the role of greenhouse gas (GHG) emissions. Furthermore, it explains the current state of science and research on climate change and its effects. Chapter 2.1.2 deals with political efforts and measures regarding the expansion of renewable energy and Chapter 2.1.3 discusses sustainability endeavors in the information and communication technology industry in particular.

2.1.1 Current State of Climate Research

Anthropogenic climate change is one of the biggest challenges of our time, at which is not only looked at since the Fridays For Future movement started and brought masses onto the streets to demonstrate for political action to limit global warming and stop the depletion of natural resources. In 1896, a distant relative of the famous Greta Thunberg, the scientist and chemistry Nobel laureate Svante Arrhenius already studied the effects of increased CO_2 concentration in the earth's atmosphere [22]. Even if these research results were not fully accurate because of several uncertainties at the time, they still show that the phenomenon of the greenhouse effect and its impact on our climate is not new to science, but well understood since around 120 years. There are several GHGs including but not limited to water vapor (H_2O), carbon dioxide (CO_2), methane (CH_4), nitrous oxide (N_2O) or ozone (O_3) [23]. These gases allow short wave radiation (light) emitted from the sun to pass through and at the same time they reflect long-wave infrared (thermal) radiation back to our solar-warmed earth [24]. [24] also states, that without GHGs, the earth's average temperature would be around -18°C, while with a pre-industrial GHG concentration we would have an average surface temperature of around 15°C. Each additional emissions change the earth's energy equilibrium, which consequently leads to a change in surface temperature. In the context of climate change mitigation, the focus of research is predominantly set on CO_2, as it accounts for about 75% of the human-generated global warming effect [25]. One of its characteristics is, that it has a lower decay rate than other heat-trapping gases, which means it remains longer in the atmosphere [26]. Furthermore, it is

the GHG with the highest positive *radiative forcing* expressed in Watt/m^2, a value introduced by the Intergovernmental Panel on Climate Change (IPCC), as a measure that indicates the change in net radiation flux based on the atmospheric concentration of a GHG (a positive value increases the energy retained by the planet and leads to warming, a negative value vice versa leads to cooling) [27].

Since the Industrial Revolution, human actions have become the main driver of global environmental change, first and foremost because of the burning of fossil fuels, which also threaten the continuance of the Holocene, an era of stability in the history of the planets environment [28]. In 2009, Johan Rockström et al. proposed the planetary boundaries framework to "define the safe operating space for humanity", broken down into nine categories. An updated version published in 2015 by Steffen et al. clearly shows the critical state, because four of the ecological systems have already exceeded their planetary boundaries, one of which is climate change [29]. The conducted analysis of the interactions between the nine planetary boundaries has shown that two of them in particular - climate change and biodiversity - have also very high impacts on the other domains. This is also why recent IPCC reports are attracting more and more the attention of policy makers, society and industry. The IPCC is at date one of the most relevant organizations in the matter of climate research. As an United Nations organization conducted of 195 member states, scientists from around the world dedicate their time to continuously assess scientific papers to provide extensive summaries of climate change impacts, future risks, as well as adaption and mitigation strategies [30]. In their most recent assessment report (AR5) from 2014, the importance of CO_2 mitigation is emphasized once again [1].

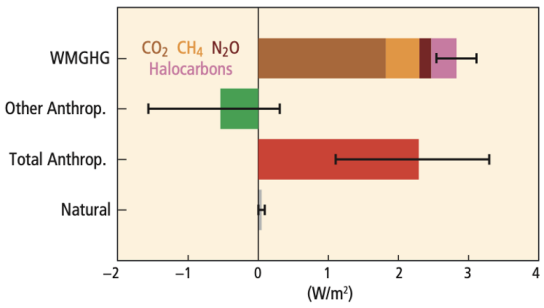


Fig. 2.1: Radiative forcing in 2011 relative to 1750, IPCC AR5 [1, p. 45]

Looking at figure 2.1, one can see a detailed breakdown of the current radiative forcing of well-mixed greenhouse gases, relative to pre-industrial times, with CO_2 being responsible for the largest share. Based on changes of the planets radiative forcing resulting from global

emissions, the IPCC developed several possible future scenarios, shown in figure 2.2. As an example, RCP 2.6 stands for "Representative Concentration Pathway with a radiative forcing of 2.6 Watt/m^2", which would be a stringent mitigation scenario, where major emission reductions must be enforced. In this case, the atmospheric GHG concentration must be stabilized at 430-480 parts per million (1 ppm $\widehat{=}$ 0,0001%) CO_2-equivalents [1, p. 22]. The other three scenarios represent two intermediate pathways and one with very high GHG emissions respectively. At the time writing, the current concentration is at an all-time high of around 420 ppm, the highest value since the beginning of the measurements [31]. Pre-industrial values averaged at around 280 ppm, with a highest previous concentration at 300 ppm [32]. According to the Mauna Loa Observatory, the annual increase of CO_2 went from roughly 0.9 ppm in the 1960s up to 2.4 ppm in the present [33], which corresponds to around 42 Gt of CO_2 emissions, assuming an preceding uptake in natural sinks like land and ocean reservoirs, which currently absorb approximately 55% of human emissions [34].

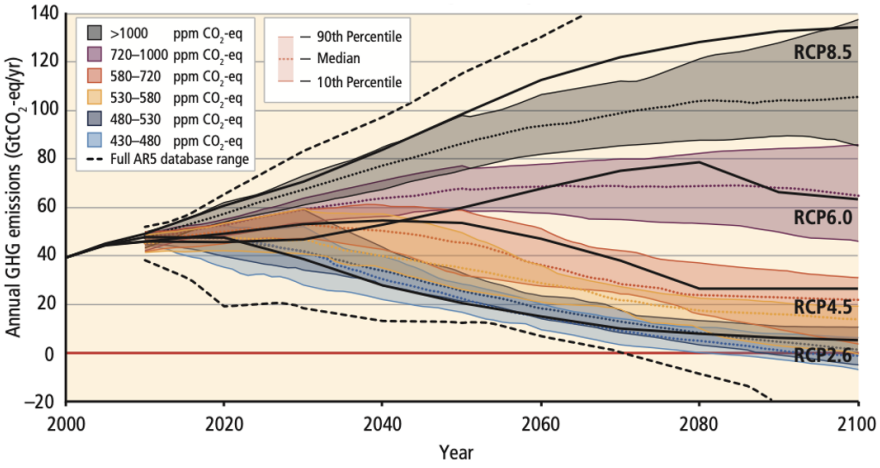


Fig. 2.2: IPCC AR5 GHG emission pathways 2000–2100 [1, p. 21]

The associated temperature rise for each of the above-mentioned scenarios and thus the resulting global risk of severe impacts are shown in table 2.1. Since not all scenarios need to be examined in detail in this thesis, the last column of the table gives an approximate level of additional risk regarding the following discussed impacts of climate change in the respective pathway.

The effects of climate change will be noticeable in each of the given scenarios. However, the intensity of these effects depends on our ability to reduce GHG emissions. With each further temperature increase, the likelihood of pervasive and irreversible impacts for people, species and ecosystems rises [35] like shown in Table 2.1. The following list represents an exemplary

Scenario	Δ°C in 2046–2065	Δ°C in 2081–2100	Risk of severe impacts
RCP 2.6	0.4 - 1.6	0.3 - 1.7	moderate
RCP 4.5	0.9 - 2.0	1.1 - 2.6	moderate to high
RCP 6.0	0.8 - 1.8	1.4 - 3.1	high
RCP 8.5	1.4 - 2.6	2.6 - 4.8	very high

Tab. 2.1: Projected Global Surface Temperature Change [1, p. 18]

subset of possible impacts that we have to face if climate change mitigation measures are not taken:

- **Environmental Risks**
  - Substantial biodiversity threats and species extinction, e.g. in marine ecosystems due to ocean acidification and warming or the loss of terrestrial ecosystems [36–38]
  - Extreme heat waves, wildfires, droughts and extreme weather events [24, 39]
  - Melting of glaciers, global sea level rise and floods [40, 41]
- **Societal Risks**
  - Global and regional food insecurity and water shortage e.g. due to reduced fisheries catch and crop failures; possibly encouraging violent conflicts [42–46]
  - Forced migration and increased human displacement [47, 48]
  - Threats to low-lying coastal areas due to flooding [49]
  - Higher human morbidity and mortality due to vector- and water-borne diseases, dehydration, heat stroke and heat exhaustion [50–52]
- **Economic Risks**
  - Immense economic damage of several trillion US-Dollars [53, 54]
  - Reduced labour productivity [55]
  - Further increase of economic inequality [56]

In summary, current climate research shows a worrying human-caused change in our ecosystems, which calls for urgent action. 82% of the fundamental ecological processes are being affected by climate change [57]. However, not only flora and fauna is threatened with extinction, but also humans are directly affected by the consequences of current developments. In general, risks are even greater for disadvantaged people in countries at all levels of development and weaker members of society like the poor, children and elderly [58]. The

devastating risks on habitats and resources harbour a lot of potential for international and regional conflicts. Climate change is quite rightly described as the biggest market failure, as we missed to internalize the cost of emissions for too long, what is now turning out to be a global burden [25]. If we take scientists seriously, it should be in the interest of everyone, be it politics, industry or individual citizens, to follow the paths of the first two scenarios proposed by the IPCC (RCP 2.6 or 4.5). The next chapter will therefore give an overview about current political efforts regarding the expansion of renewable energies in the world and Germany.

2.1.2 Renewable Energy Transition

As early as the 1990s, more than 150 countries met and signed the first international environmental agreement with the aim of combating anthropogenic climate change. The United Nations Framework Convention on Climate Change was signed in 1992 at the United Nations Conference in Rio de Janeiro and entered into force in 1994 [59]. The now 197 Parties to the Convention meet annually at a world climate summit to define measures and strategies and to conclude multilateral agreements [60]. In 2015, along with other important landmarks in international policy shaping, like the introduction of the 17 Sustainable Development Goals at the UN Sustainable Development Summit in New York, the famous so called Paris Agreement, a legally binding international treaty on climate change has been passed [61]. According to the Paris Agreement, all member states are obliged to submit nationally determined contributions (NDCs), which are actions to reduce their greenhouse gas emissions [62]. These voluntary commitments are intended to contribute to the jointly formulated goal of limiting the temperature increase to well below two degrees, if possible to 1.5 degrees. As of now, 190 states, including important signatories like the US, China, Russia, India or the EU-28, representing about 97% of global greenhouse gas emissions, have ratified the Agreement [63]. In game theory, an international cooperation like the Paris Agreement is often depicted as a prisoners' dilemma, whereas the states are better off reducing their emissions together, but mostly economically justified self-interests impels them to keep on emitting [64]. However, low-carbon solutions are becoming competitive across sectors, e.g. the power sector [62], which can lead to economic benefits, even in the absence of collective climate action [65].

One long-term project to reduce emissions in Germany is the so called energy transition, which refers to a sustainable way of generating and using energy by turning away from traditional electricity and heat generation. Up to now, conventional or fossil energy sources such as coal, natural gas and crude oil have been used to a large extent for the generation of energy. Renewable energies, for example wind power, photovoltaics and biogas are yet to

Energy Source	CO_2-equivalent in g/kWh	Cost* per kWh in € Cent
Nuclear	12	4
Wind	86	9.1
Solar Park	143	8.2
Solar Roof	143	12.2
Bio Gas	230	13.3
(Brown) Coal	1150	10.4

*Cost including system cost and a CO_2 price of 50€/t

Tab. 2.2: Comparison of selected energy sources by emissions and price [66, 67]

become the most important energy source in the long term as part of the energy transition. In 2020, 502.6 TWh power were generated in Germany, which corresponds to around 739 million tonnes of total greenhouse gas emissions (fluctuating between 87 and 664 g per kWh) with a renewable energy share of 35%:

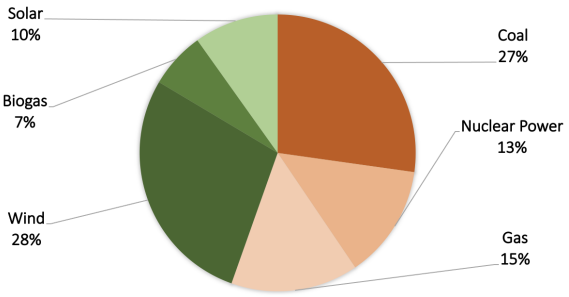


Fig. 2.3: Power Generation by Source, Germany 2020[68]

In Germany, the Climate Protection Plan 2050 has set the goal of achieving greenhouse gas neutrality by 2050 [69]. In addition, the plan contains concrete sectoral targets, which were laid down in the Federal Climate Protection Act passed in 2019 and became legally binding. At the end of 2019, the German government also published the new Climate Protection Programme 2030, which includes measures like CO_2 pricing and the expansion of renewable energy up to a share of 65% until 2030 (e.g. through mandatory solar installation for all new buildings) as central approaches to mitigate climate change [70]. Furthermore, the EU Commission presented the EU Green Deal in 2019, which aims to achieve a renewable energy share of 40% in the EU energy mix by 2030 and complete greenhouse gas neutrality in the EU by 2050 [71]. One concrete measure to reach these goals is the EU Emissions Trading System (EU ETS), which covers around 40% of the EU's greenhouse gas emissions, especially in the power sector, manufacturing industry and aviation sector [72]. The overall volume of greenhouse gases that can be emitted is limited by a cap on the number of emission allowances, which can be traded between polluters within the system. One allowance gives the right to emit one tonne of

carbon dioxide or the equivalent amount of other greenhouse gases. For 2021, the EU wide cap is fixed at 1.571.583.007 allowances (≙1.57 Gt CO_2 emissions) [73]. In 2019, the EU-wide total emissions were at a level of ~5.57 Gt CO_2 (40% ≙ ~2.23 Gt CO_2) and the worldwide emissions at a level of ~38 Gt CO_2-equivalents [74]. This cap and trade system aims to promote the use of low-carbon technology and renewable energy through economic incentives, as the number of allowances is becoming less at an increased annual linear reduction factor of 2.2% and thus the price per allowance rises if reductions cannot be realised at the same speed [73]. Similar trading schemes have also been implemented in other countries like China and on a federal level in the USA [75], which makes the expansion and use of renewable energies not only ecologically but also economically interesting for companies and industries all over the world.

2.1.3 Sustainability in Information Technology

As a result of the corresponding laws and the societal discourse, many industries are undergoing great transformations to adapt to new market conditions. Many initiatives like research around the topic of Green IT [76], the Climate Neutral Data Centre Pact[1] or different approaches to sustainable software engineering also show an intrinsic motivation of the information and communication technology industry to commit more strongly to the active mitigation of climate change. Microsoft for example is working towards carbon negativity until 2030[2] and offers free educational resources for developers to integrate cross-domain knowledge of climate science, software, hardware, electricity markets, and data center design into daily processes[3]. New technologies which support proliferation of renewable energy are on the rise and accelerate the implementation of trends like decentralized generation of renewable energy and local power storage assets [77]. Furthermore, a PwC study found an approximate increase of 3750% in climate tech investments between 2013 and 2019, with a continued interest from big tech companies like Amazon (Climate Pledge Fund) or Microsoft (Climate Innovation Fund) [78]. Tech companies are also not just on the front line of green tech investments, but also need to undertake efforts to make their own businesses more sustainable [79]. Due to their sheer size, cloud computing providers in particular have a lot of levers at their disposal to influence the entire lifecycle of hardware and software in the long term through innovation in different areas such as the implementation of more efficient cooling techniques, on/off-site renewable energy generation or workload migrations [80]. A recently updated prediction on the electricity footprint of the information and communication technology

1 https://www.climateneutraldatacentre.net/, *accessed 2022-01-03*
2 https://www.microsoft.com/de-de/nachhaltigkeit, *accessed 2022-01-03*
3 https://docs.microsoft.com/en-us/learn/modules/sustainable-software-engineering-overview/, *accessed 2022-01-03*

industry [81], which is expected to increase from currently 1% to up to 9% of global electricity consumption in 2030 [82], also clearly shows the importance of technologies that enable an efficient integration of renewable energy into information systems to stay in line with net zero emissions efforts. Two relatively new initiatives implementing renewable-aware load shifting have been done by Google[4]. As on of the first cloud providers, Google already runs it's data centres on 100% (purchased) renewable energy. However, their goal is to decarbonize their electricity use completely around the clock by coupling their data centers more closely to available carbon-free energy sources like solar and wind. A pilot they did in 2020 is a so called carbon-intelligent computing platform, which utilizes time-based load shifting to place flexible workloads to times of the day when renewable energy supply is high. The second approach exploits location-based availability of renewable energy at different data center locations on a global scale using day-ahead predictions of how heavily a certain grid will be relying on carbon-intensive energy. The direction of future developments is clearly set by industry leaders like Google and it once again shows the important role such technologies will play soon to stay competitive.

2.2 Distributed Systems

The following chapter provides an introduction to distributed systems, especially in the field we refer to as cloud computing. Chapter 2.2.1 shows general concepts of cloud computing including different manifestations such as fog or edge computing. Furthermore, a brief introduction to containers and their importance for today's application landscape is given in Chapter 2.2.2, which leads us to the reason why Kubernetes is given as the enabling system in this thesis.

2.2.1 Cloud Computing

In general, the National Institute of Standards and Technology (NIST) defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [83], whereas various delivery models like Infrastructure, Platform or Software as a Service are possible to be chosen by the customer [84].

4 https://blog.google/outreach-initiatives/sustainability/carbon-aware-computing-location/, *accessed 2022-01-03*

Cloud computing services are mainly offered by specialized public, private or hybrid cloud providers like Amazon Web Services or Microsoft, who operate massive data center locations distributed across the globe [85]. The market dominating AWS[5] cloud for example consists of 22 so called regions, which are defined as locations with multiple data centers. A logical group of data centers is called an Availability Zone and each AWS Region consists of multiple isolated and physically separate and independent Availability Zones within a geographic area. In addition, AWS offers so-called Local Zones, i.e. locations that are closer to the end users. Putting this setup in the context of this thesis, one can observe that it is ideally suited for location-based renewable energy use at different sites that are located within a logical network. Similar structures are also in place at other cloud providers[6][7]. However, what is commonly known as cloud computing can be further divided into cloud, fog, and edge computing paradigms, each coming with their own characteristics. Especially for emerging IoT use cases with an increasing data volume and velocity generated by edge devices, the cloud might not be able to cover all requirements [86].

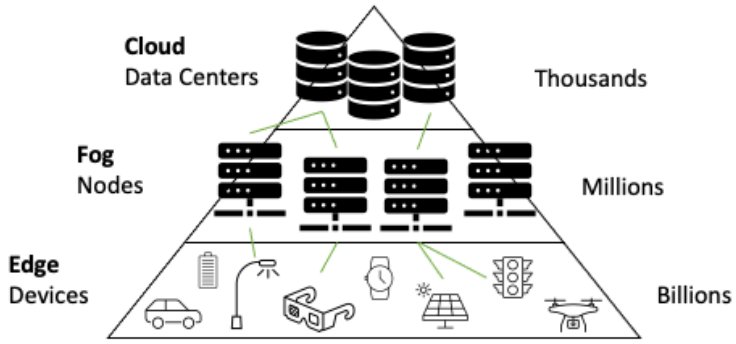


Fig. 2.4: Cloud Computing Paradigms, adapted from Qayyum et al. [87], Figure 1

In Figure 2.4 one can see the evolution from cloud to edge computing. To solve issues of high-bandwidth, geographically-dispersed, low-latency and privacy sensitive applications on the edge, the concept of fog computing was introduced to bring computation and data processing closer to the end user [88]. According to the Cisco Annual Internet Report, there will be 29.3 billion networked devices by 2023, up from 18.4 billion in 2018 [89], which shows that the need for digital services is growing and the cloud is already used by billions. The increasing number of nodes for fog and edge computing, networked and geographically distributed, thus offers another potential application area for the present work.

5 https://aws.amazon.com/de/about-aws/global-infrastructure/regions_az/, *accessed 2022-01-03*
6 https://cloud.google.com/about/locations, *accessed 2022-01-03*
7 https://docs.microsoft.com/en-us/azure/availability-zones/az-overview, *accessed 2022-01-03*

2.2.2 Containers

Like we just learned before, resources in cloud computing are shared among their users to achieve efficiency gains in terms of overall hardware utilization. To achieve this, virtualization is used. Therefore, a so called hypervisor software creates an abstraction layer over the physical hardware of a server, which then acts as an interface to the virtual machines. Resources like CPU, memory or network can then be easily managed and shared across VMs, while still being completely isolated from each other. Each VM runs its own operating system and behaves like an independent computer [90].

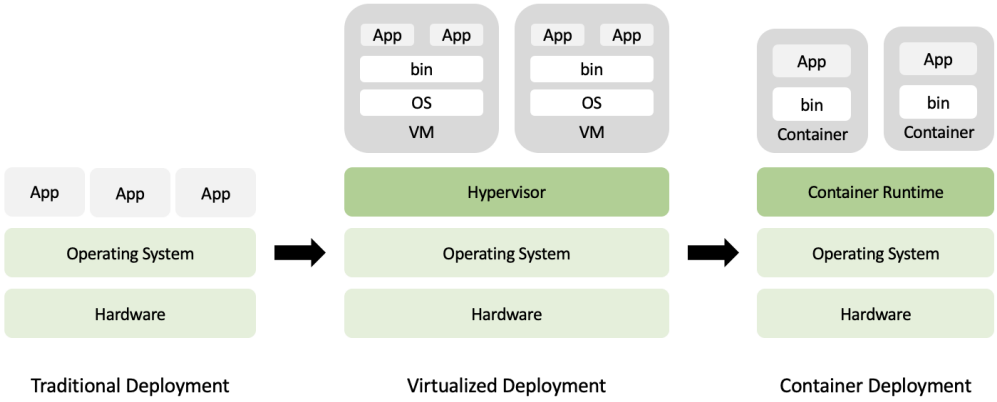


Fig. 2.5: Evolution of deployments, adapted from Kubernetes [91]

Unlike classic virtual machines, containers do not emulate their own operating system, but use that of the host system like illustrated in Figure 2.5, which makes them much more lightweight than VMs [91]. All files, configurations, dependencies and libraries required for execution are packed into a container image. Container images are single files and become containers as soon as they are executed, thus they are easily transferable to other host systems and developers can move applications between different environments with minimal effort. Among several container runtimes, Docker Engine, containerd and LXC are the most famous. Containers are often used to package single applications or even functions, what makes them a popular choice to create modern microservice architectures, which also favor the use of DevOps techniques like CI/CD [92]. This is also confirmed by a survey by the Cloud Native Computing Foundation, which found a 300% increase in the use of containers in production systems between 2016 and 2020 [93]. As those cloud-native microservice architectures can consist of hundreds of containers working together, a manual management would cause disproportional overhead. To avoid this, the whole system has to be orchestrated somehow to automate tasks like provisioning, resource management and scaling resiliently. Therefore,

various container management systems like Docker Swarm, Nomad, AWS Fargate, Apache Mesos or Kubernetes come to the rescue. Out of the aforementioned, Kubernetes is the clear industry leader by far, which is confirmed by several surveys [93–95]. Therefore, it is well suited to reach the widest possible audience through solutions tailored to the system, such as renewable-aware scheduling.

2.3 Kubernetes

As Kubernetes represents the technological core this thesis is building on, an introduction to the background of Kubernetes is provided in the current chapter. The following chapters will provide a detailed explanation of Kubernetes components, its object model as well as a deep dive in the scheduling process.

According to the official documentation, Kubernetes is a "portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation"[91]. It is written in Go and was mainly created by Google engineers as a successor of two different container management systems. One was Borg, a system to manage long-running services and batch jobs, and the other was Omega, a system similar to Borg, but with a more consistent and principled software architecture and other improvements taken from Borgs lessons learned [96]. In 2014, Kubernetes was donated to the Cloud Native Computing Foundation, which serves as a vendor-neutral home for many open source projects.

Kubernetes' architecture design follows decoupled microservices patterns and can be extended by writing custom resources, operators, APIs, scheduling rules or plugins. Kubernetes aims to support a variety of workloads, including stateless, stateful, and data-processing workloads. The platform offers several features to operate containerized applications in a distributed environment. It automatically schedules containers on nodes based on resource needs and constraints, to maximize utilization without sacrificing availability. Containers from failed or unsuitable nodes are replaced and rescheduled to appropriate nodes. It redistributes traffic and restarts containers that do not pass health checks. Depending on CPU utilization or other metrics, apps can be scaled manually or automatically to match service requirements. Furthermore, DNS names are assigned to sets of containers to enable load-balancing across service containers (e.g. multiple nginx instances). Application updates (e.g. new versions of container images) or configuration changes can also be applied without any downtime, ensured by the systems rollout and rollback features. Data that needs to be passed into the

application is handled separately from the container image, in order to avoid a re-build of the respective image and exposure of the stack configuration in version control repositories. In addition to container orchestration, Kubernetes also takes care of mounted storage respectively. This can include software-defined storage connected from local storage, external cloud providers, distributed storage, or network storage systems. Futhermore, batch execution and long-running jobs are supported as well, which makes Kubernetes also a good choice for machine learning tasks like model training or batch inference[8]. Recent developments of projects like K3s[9], KubeEdge[10] or MicroK8s[11] also indicate a need of lightweight Kubernetes distributions for fog and edge computing environments.

2.3.1 Components

A Kubernetes cluster consists of one or more worker nodes, as well as a one active and optional synced fail-over master nodes. The master node provides a running environment for the control plane, which manages the state of the cluster and its workload resources on the worker nodes. A worker node provides a running environment for client applications, which are encapsulated in pods. A pod is the smallest unit in Kubernetes, consisting of one or more containers scheduled together. The essential components for a complete working cluster are also depicted in the architectural schema below and will now be discussed.

kube-apiserver

The API server is running on the master node and coordinates all administrative tasks by intercepting calls from any clients and agents, which call the interface either manual (e.g. via kubectl in the CLI) or programmatically via client libraries. During validation and processing of incoming calls, the API server communicates with the etcd data store, to retrieve and update the clusters state information on demand. The Kubernetes REST APIs that are exposed by the API server can be divided into three independet group types: Core Group (/api/v1), which includes objects such as Pods, Services, Nodes, Namespaces or ConfigMaps. The Named Group (e.g. /apis/scheduling.k8s.io/v1) and the System-wide Group (e.g. /healthz). These apis are also used later to intervene in the scheduling process. Like other control plane components, the API server is highly customizable and scalable.

8 https://mlinproduction.com/k8s-jobs/, *accessed 2022-01-03*
9 https://github.com/k3s-io/k3s, *accessed 2022-01-03*
10 https://github.com/kubeedge/kubeedge, *accessed 2022-01-03*
11 https://github.com/ubuntu/microk8s, *accessed 2022-01-03*

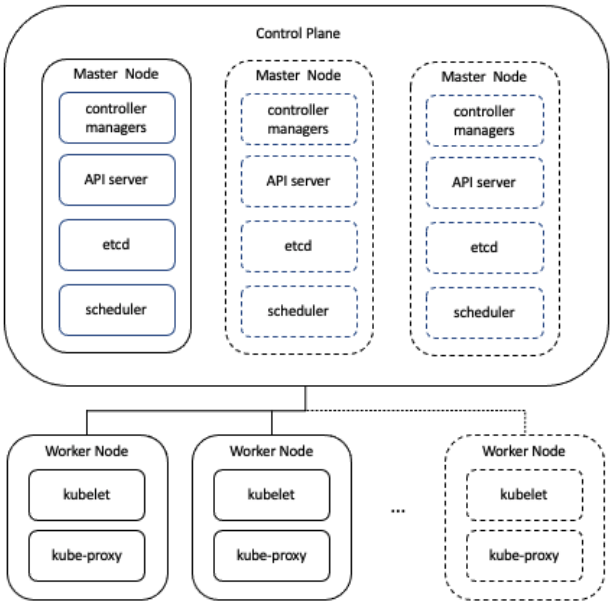


Fig. 2.6: Kubernetes components [91]

kube-scheduler

The kube-scheduler is one of the key components and of particular importance for this work. It is responsible for assigning new workload objects to nodes. Beneath the clusters state and the objects requirements, the kube-scheduler takes several conditions like Quality of Service requirements, data locality, node affinity, node anti-affinity, taints or toleration into account to make a scheduling decision. The scheduler basically runs through two main steps - filtering and scoring. First, the scheduling algorithm isolates the node candidates which match the filter criteria and then passes them on to the prioritization, which consequently ranks each valid node. The result of the scheduling decision is communicated back to the API server, which then manages the workload deployment in cooperation with other control plane components. The scheduler can be configured through scheduling policies and profiles, as well as custom schedulers and extensions.

kube- and cloud-controller-manager

The controller managers are responsible to regulate the clusters state by continuously comparing the desired (provided by configuration data) and current state (obtained from etcd data store). The kube-controller-manager takes action when nodes crash, pods counts are not as expected or endpoints, etc. have to be created. The cloud-controller-manager interacts with the infrastructure of the respective cloud provider.

etcd

Etcd is a strongly consistent, distributed key-value data store used to persist a clusters state and configuration data like ConfigMaps or Secrets. It can be run on a master node or on its dedicated host. Configuration data is always only appended, but never replaced in the data store. Like mentioned before, the only control plane component that is able to communicate with etcd is the API server. To ensure high availability, etcd is usually replicated in production environments.

kubelet

The kubelet agent runs on each worker node and communicates with the control plane components. It is responsible to manage pods, it receives pod definitions from the API server, monitors health and resources and interacts with the container runtime inside pods.

kube-proxy

The kube-proxy acts as a network agent on each node and provides dynamic updates for all networking rules. Furthermore, it abstracts the pods networking and forwards TCP, UDP, and SCTP requests to pods. The kube-proxy is especially important for microservice architecture, as they rely heavily on networking features such as container-to-container communication inside pods, pod-to-pod communication on the same node and across cluster nodes or external accessibility through services.

2.3.2 Object Model

The Kubernetes object model represents different persistent entities in the cluster, which describe for example the containerized apps that are running, the nodes where they are placed on, resource consumption of any application or policies attached to the applications. The control plane uses these entities to continuously match the object's actual state to the object's desired state, e.g. in case node failures or updates to pod configurations. Objects are most often provided in YAML format, which is converted into a JSON payload and sent to the API server. The state of objects can be easily updated with client libraries or the kubectl command line tool, e. g. by using the command taking the new descriptive state in the file object.yaml as an input: `kubectl apply -f object.yaml`. A relevant subset of those objects will now be introduced.

Pod

A Pod is an environment to run a group of one or more containers and represents the smallest deployable unit in a cluster, each with its own IP address. Instead of managing containers directly, Kubernetes only tracks the state of containers through kubelet and restart them in case of a failure. A Pod can thus be seen as a wrapper around the application. Everything inside a Pod shares a context and its contents are always co-located and co-scheduled on the same logical host (or on the same VM in a cloud context). If multiple applications exist in one Pod, they are usually tightly coupled. However, the one-container-per-Pod model is the most common. One pod contains a single application instance and can be scaled out by horizontal replication across the cluster. Since Pods are rather ephemeral and non-permanent, workload resources are usually used to create and manage Pods on the cluster, which for example has the advantage that in case of a node failure, Pods are automatically restarted to restore the desired cluster state given by the respective configuration. One of these aforementioned workload resources is a so called Deployment.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    role: myrole
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
```

Fig. 2.7: Exemplary nginx Pod object's configuration manifest in YAML format

Deployment

Deployment objects provide declarative updates to Pods and ReplicaSets (used to maintain a stable set of replica Pods running at any given time). It allows for seamless application updates and scaling. Furthermore, the Deployment offers the handy feature that only a certain number of Pods are down while they are being updated. By default, it ensures that at least 75% and at most 125% of the desired number of Pods are up (25% max unavailable / surge). Deployment object - as any other object are most often provided in a YAML format, which is converted by kubectl into a JSON payload and sent to the API server.

Namespace

Namespaces are mainly used to isolate teams, tiers or applications. Kubernetes creates four Namespaces out of the box: kube-system, kube-public, kube-node-lease, and default. The most relevant for us are kube-system, which contains mostly control plane agents like the kube-scheduler and the default Namespace, where objects (e.g. Deployments) are created by default unless another Namespace is provided. Within Namespaces, resource limits can be set.

Service

Service objects are used to abstract the communication between applications (microservices) running in the cluster, or with the external world. A Service offers a single DNS entry (Service name) for each application managed by the Kubernetes cluster, regardless of the number of replicas, by decoupling the load balancing access point from the set of coherent Pods. To define this set, Services use Labels and Selectors that can be specified in its object YAML - for example app==frontend, which selects all Pods with the respective Label and logically groups them together. This is very handy if different applications have to talk to each other, but cannot keep track of every disposable IP address of their respective Pods. Depending on how the access scope is defined, a suitable ServiceType must be selected. This then decides, whether certain services can only be accessed within the cluster, only from outside, both or any other possible configuration.

ConfigMap

A ConfigMap is used to decouple environment-specific configuration from container images, so that configuration data is set separately from application code, e.g. for database host names in different environments or development stages. It can store non-confidential data in key-value pairs - for sensitive data so called Secrets are used. If they are in the same Namespace, the reference to a ConfigMap can be defined in a Pods spec section. A Pod can consume a ConfigMap in four different ways: as container command and args, as environment variables for a container, readable from a volume and via the Kubernetes API.

Object Metadata

In addition to the object configuration itself, Kubernetes also offers the possibility to attach metadata of different kind such as **labels** or **annotations** in the form of key-value pairs to objects. Labels are intended to specify identifying attributes or to organize and select subsets of objects, for example to distinguish which environment a pod should serve one can add

a *"environment" : "dev"* label or to point out certain node equipment one can add labels like *"accelerator": "nvidia-tesla-V100"* [12]. Annotations on the other hand are used to attach non-identifying metadata to objects[13]. The data in an annotation can be structured or unstructured, like contact data, URLs, image hashes or other arbitrary information. Labels and annotations can also be used in the node selection process of pod scheduling, e.g. to exclude some nodes if pods to be scheduled need certain hardware, or to prioritize some nodes over others based on external parameters.

2.3.3 Scheduling

According to Bittencourt et al. Scheduling is a decision-making process that enables resource sharing among a number of activities by determining their execution order on the set of available resources [97]. In Kubernetes, the kube-scheduler allocates Pods (activities) to Nodes (available resources) considering hardware and software requirements. After Pods are created and need to be scheduled, they are added to a waiting queue. The kube-scheduler continually monitors this queue and looks for so called feasible nodes that fit the Pods requirements by also being aware of each nodes current state. The scheduling cycle consists of nine stages (Sort, PreFilter, Filter, PostFilter, PreScore, Score, NormalizeScore, Reserve and Permit), followed by the binding cycle, which notifies the kube-apiserver about the final scheduling decision.

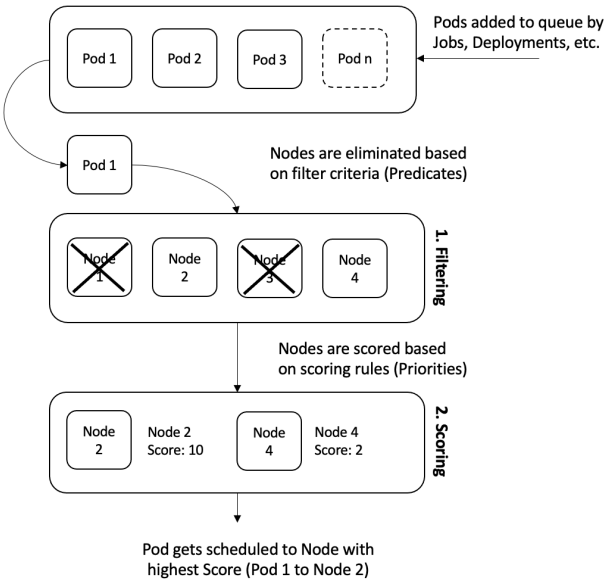


Fig. 2.8: Filter and Score Stage in a Pod Scheduling Lifecycle

12 https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/, *accessed 2021-09-09*
13 https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/, *accessed 2021-09-09*

Two of the stages are particularly important for the selection of suitable nodes and therefore represent the core of the scheduling process: **Filter** and **Score**. To configure the filtering and scoring behavior of the scheduler, one can utilize two options: **Scheduling Policies** and **Scheduling Profiles**. A Scheduling Policy can be used to specify the predicates and priorities that the kube-scheduler runs to filter and score nodes, respectively. In the filtering stage, the scheduler checks all configured predicates - hard constraints that cannot be violated - for instance the need for specific node labels (e.g. disk==ssd) or a certain amount of memory available[14]. After this stage, nodes that do not meet these requirements are eliminated from the set of feasible nodes, which is then passed to the scoring stage. In this stage, priorities are considered. These are soft constraints, for example an even distribution of workload across nodes, that can be weighed differently and violated if needed. Subsequently, the Pod to be scheduled is assigned the node with the highest priority. The built-in Predicates and Priorities are diverse and useful to configure the scheduler already to some extend to ones own needs. They can be optionally added to the schedulers ConfigMap YAML. In case of the priorities, further customization can be applied by assigning weights to each policy. In the scoring stage, nodes are given a score between 0 (worst) and 10 (best). The final score of each node is calculated by adding up all the weighted scores. However, metrics that can be used to adapt scheduling decisions are by default only cluster internal, mostly resource based technical measures. As an example, following selected predicates and priorities are supported[15]:

Predicates	Priorities
PodFitsResources checks if the node has free resources to meet the requirement of the Pod. ***MatchNodeSelector*** checks if the node selector matches the node labels. ***PodFitsHostPorts*** checks if a node has free ports which the Pod is requesting. ***NoDiskConflict*** checks if a Pod can fit on a node due to the volumes it requests.	***LeastRequestedPriority*** favors nodes with fewer requested resources. ***MostRequestedPriority*** favors nodes with most requested resources. ***ImageLocalityPriority*** favors nodes that already have the container images cached locally. ***BalancedResourceAllocation*** favors nodes with balanced resource usage.

Tab. 2.3: Selected Default Scheduling Policies

Just like Policies, Scheduling Profiles can be used to extend all scheduling stages by activating so called plugins that implement the stages respective interfaces. Some of those plugins are already enabled by default, giving the kube-scheduler its standard behaviour, e.g. the *NodeUnschedulable* plugin (extends Filter), which filters out nodes that have *.spec.unschedulable* set to true. However, Scheduling Profiles and Scheduling Policies can both be used for

14 https://www.youtube.com/watch?v=rDCWxkvPlAw, *accessed 2022-01-03*
15 https://kubernetes.io/docs/reference/scheduling/policies/, *accessed 2021-08-26*

configuration and offer quite similar, sometimes even identical functions - but each in its own form as a plugin or predicate / priority. However, at the time writing, the feature state of Scheduling Profiles is not stable and might therefore only be included in the next release. Possibilities to customize the behavior of the scheduler with own profiles and policies are discussed more in-depth in Section 4.3.1 as part of the scheduler extension method selection.

3 Related Work

An extensive review of relevant literature resulted in a clear picture regarding the topicality of the subject area being worked on. The overall relevance of the topic of the effective use of renewable energies also in the field of cloud computing is reflected in the number of authors dealing with it. Also, the modification of the kube-scheduler is a common approach to improve the behavior of Pod placements in Kubernetes clusters and thus make the system more efficient.

Wojciechowski et al. [98] are extending the default Kubernetes scheduler to include network metrics awareness in the process, using metrics collected from Istio Service Mesh. Huaxin et al. [99] combine existing scheduling alrogirthms of Microsofts HiveDScheduler with default Kubernetes policies supplemented by resource metrics like GPU usage to improve multi-tenant resource usage in a cluster. A scheduler that takes fair distribution of resources among users of a multi-tenant cluster into account is also presented by Beltre et al. [100]. Santos et al. [101] implement a network-aware scheduler extender to reduce network latency in smart city environments. Another network-aware approach is presented by Rossi et al. [102] to improve application latency in geo-distributed clusters. With their Gaia Scheduler Song et al. [103] achieve an improved use of GPU resource allocation across Kubernetes clusters by taking the network topology and the resulting resource access cost tree into account. Yang et al. [104] also aim to increase the utilization of cluster resources, by optimizing the scheduling algorithm with neural network prediction methods. Fu et al. [105] address scheduling for short-lived applications by monitoring their progress and estimate their completion time to balance resource contention on the workers. Townend et al. [106] create their own scheduler which takes software and hardware models of a data center into account to improve its efficiency. Katenbrink et al. [107] present an approach for dynamic scheduling in IoT environments, considering mobility and heterogeneity of nodes. Also located in the field of IoT and Edge computing is the work of Ogbuachi et al. [108], who integrate real-time information about the edge devices into their scheduling process to be able to react quicker to environmental changes. Rausch et al. [109] complement the default scheduler with several priority functions which take trade-offs between data and computation movement into account to facilitate the use of Kubernetes in Edge scenarios. A QoS-oriented scheduling with a focus on deep learning

jobs with deadlines is provided by Xing et al. [110], who modify the default Kubernetes scheduler by considering the estimated training time to improve the completion rate of jobs. Different approaches to optimize the use of renewable energy or to minimize the carbon footprint of systems are presented in [6–19], where none is directly related to the modification of Kubernetes components. Some of the aforementioned authors also do not even offer a scheduling-based solution, but achieve improved utilization of renewables through other approaches, such as load balancing optimization, which why these works are not introduced in further detail.

Author(s)	Renewable Awareness	Forecasts	K8s	Scheduling	Descheduling
Wojciechowski et al. [98]	○	○	●	●	○
Huaxin et al. [99]	○	○	●	●	○
Beltre et al. [100]	○	○	●	●	○
Santos et al. [101]	○	○	●	●	○
Rossi et al. [102]	○	○	●	●	○
Song et al. [103]	○	○	●	●	○
Yang et al. [104]	○	○	●	●	○
Fu et al. [105]	○	○	●	●	○
Townend et al. [106]	○	○	●	●	○
Katenbrink et al. [107]	○	○	●	●	●
Ogbuachi et al. [108]	○	○	●	●	○
Rausch et al. [109]	○	○	●	●	○
Xing et al. [110]	○	○	●	●	○
James et al. [20]	●	○	●	●	●
Kaur et al. [21]	●	○	●	●	○
Grange et al. [6]	●	●	○	●	○
Deng et al. [7]	●	○	○	●	○
Qu et al. [8]	●	●	○	●	○
Chen et al. [9]	●	●	○	●	○
Kumar et al. [10]	●	○	○	●	○
Goiri et al. [11]	●	●	○	●	○
Xu et al. [12]	●	○	○	●	○
Zhou et al. [13]	●	○	○	○	○
Han et al. [14]	●	○	○	○	○
Wang et al. [15]	●	○	○	○	○
Li et al. [16]	●	●	○	○	○
Fan et al. [17]	●	○	○	○	○
Guo et al. [18]	●	○	○	○	○
Khosravi et al. [19]	●	●	○	○	○
green-k8s-(de)scheduler	●	●	●	●	●

Tab. 3.1: Literature review of related fields of interest

However, the approaches are diverse and can be delimited from this thesis by looking at its different subtopics. Those subtopics are defined as: 1. The consideration of renewable energy availability or carbon footprint of given systems, 2. The consideration of energy forecast at scheduling time, 3. Kubernetes as underlying system and enabler, 4. The modification of

scheduling features or algorithms and 5. The consideration of adaptive re- or descheduling due to changing conditions. On the basis of these topics, the overview in Table 3.1 could be drawn up, while each work overlaps with at least one subtopic (● means the subtopic is covered and vice versa).

A closer look should be taken at [20] and [21], which are to my best knowledge the only ones who discuss improving the integration of renewables by modifying the Kubernetes scheduling behavior. While James at al. [20] discuss the Kubernetes scheduler and its extensibility in their paper, it rather suggests the use of a custom global scheduler that selects a region with a low carbon intensity for the creation of a whole new cluster independently of the scheduling process of the internal kube-scheduler, which does not affect the scheduling for Pods at the node level. Furthermore, unlike the green-k8s-scheduler, the low carbon scheduler of James et al. does neither consider energy forecasts nor Pod runtimes, which makes it only suitable for workloads that are not sensitive to interruptions. Kaur et al. [21] provide a multi-objective scheduler that is optimizing scheduling decisions to minimize interference and the carbon footprint on nodes. Similar to this thesis, the available renewable energy output that is available at the nodes at the time of scheduling is used as one of the decision parameters. However, the KEIDS scheduler does not take any forecasting into account and therefore also not the volatility of renewable energy sources, which may affect their overall utilization significantly. Furthermore, no adaptive rescheduling of workloads is considered if the underlying energy shares change.

4 Conceptual Approach

4.1 System Design

The goal of this thesis is to extend Kubernetes with the functionality to place Pods (workloads) at Nodes where renewable energy is available and to maximize its overall utilization, considering the nodes power consumption. Furthermore, adaptive rescheduling should enable the consideration of changing external influences such as fluctuating wind speeds or solar irradiation.

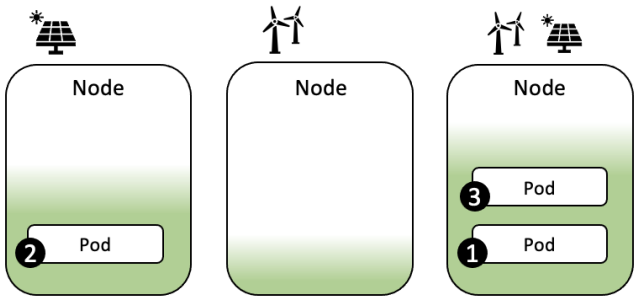


Fig. 4.1: Concept of Renewable Aware Scheduling

Three components need to be developed in order to enhance a cluster with renewable aware scheduling capabilities:

- **green-k8s-scheduler**
- **green-k8s-descheduler**
- **green-k8s-energy-management**

The green-k8s-energy-management is meant to be a substitute for an energy management system, such as smart meter data management systems, which can provide the full range of generation, consumption and forecasting data in a productive use-case to allow for finer configuration and the integration of real data. Most data centers and industrial buildings are already equipped with such systems nowadays, as it offers valuable insights to improve the overall energy efficiency of sites and buildings as well. In this case, it provides the data

which is written into the node annotations. The green-k8s-scheduler is an extension of the default kube-scheduler, responsible to extend the scoring phase with energy data gathered from the nodes. It implements a specially designed scoring algorithm to find the best fit for each Pod placement. On a regular basis, the green-k8s-descheduler checks if Pods are still placed on the most sustainable nodes and starts an eviction process if this is not the case. To do so, it uses a similar scoring logic as the scheduler, enabling adaptive changes of the system. The components that were just introduced will be described in more detail hereafter. Each of these components is available on GitHub[1][2][3]. The components are highlighted in green in the depicted architecture diagram.

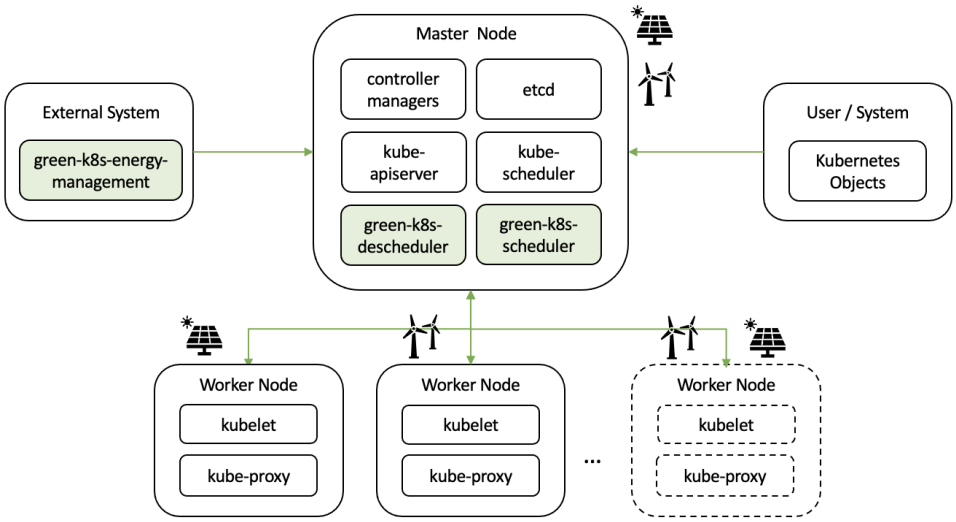


Fig. 4.2: Conceptual System Architecture

In the scenario at hand, each node represents a unique geographic location in a distributed cluster. This setup is conceivable in various manifestations of fog or - in a smaller scale - even edge computing scenarios, for example in smart city applications in which multiple distributed fog locations process data from nearby sensors or edge devices before sending it to a centralized cloud data center, or in clusters that span multiple locations of a company, each equipped with its own nodes and renewable energy appliances respectively.

1 https://github.com/7imo/green-k8s-energy-management, *accessed 2022-01-02*
2 https://github.com/7imo/green-k8s-scheduler, *accessed 2022-01-02*
3 https://github.com/7imo/green-k8s-descheduler, *accessed 2022-01-02*

4.2 Energy Management System

Energy data management, monitoring and analysis are an important part of improving the energy efficiency and performance of residential and industrial buildings or sites [111]. Especially in data centers, energy data is already used in approaches to improve the integration of renewables, offer demand response capabilities or to optimize heating and cooling [112]. The consumption, production and weather data available in such systems is also utilized in the present work to calculate the renewable energy excess available in the present and near future. Data from energy management or intelligent metering systems can easily be made available internally at each cluster node via the kube-apiserver in order to be included in the scheduling process as depicted in Figure 4.3.

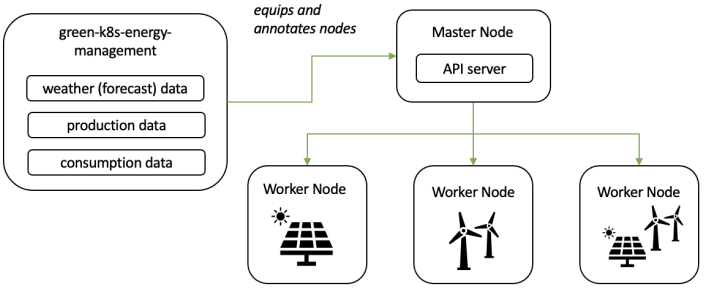
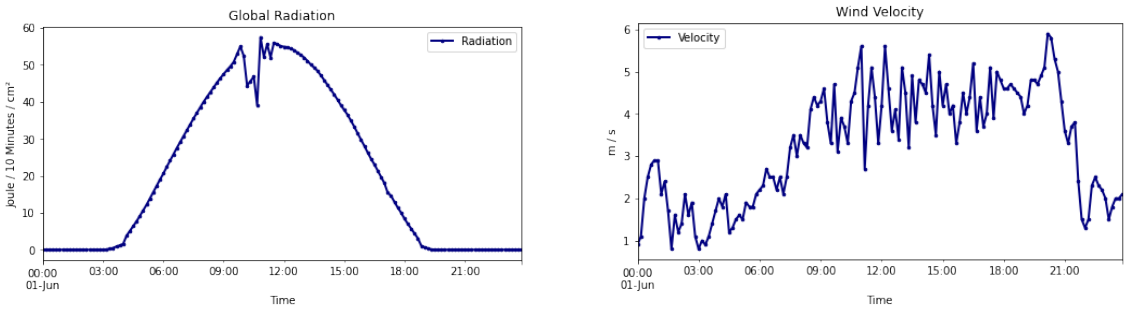


Fig. 4.3: Node Equipment and Annotation through the Energy Managment System

The *green-k8s-energy-management* will generate data to simulate locally equipped renewable energy plants, such as solar panels or wind turbines. Depending on the equipment, the respective electricity production data will be constantly annotated to each node. In order to consider the volatile nature of renewable energies like shown in Figure 4.4 as well as possible, the data will consist of 25 parameters per annotation interval: one 10-minute "real-time" output (which can also be more granular depending on the data available) and hourly forecasts from 1 hour to 24 hours ahead, containing the respective average renewable energy availability at the node for each period, e.g. the average availability between now and in 24 hours for the 24 hour data. Furthermore, the current energy consumption of a node should be written into the annotation as well in order to be able to calculate the remaining renewable energy excess and its resulting node score, which will be described in detail in section 4.3.3.

Forecasting

The field of time series analysis and forecasting in general and renewable energy forecasting in particular already covers a wide range of research areas, especially in stochastic modeling and



(a) Global Radiation **(b)** Wind Velocity

Fig. 4.4: Volatile Availability of Renewables

machine learning. Various forecasting methods are available to provide valuable information about the expected changes in the availability of renewable energy in the near future. Accurate forecasts can help to improve scheduling decisions to enable profound forward-looking Pod placements, that do not only consider current renewable energy availability, but also include different time horizons. This can help to keep the quality of service stable in the long term, e.g. to avoid frequent reallocation of Pods to intermediate better fitting nodes, as not only real-time measurements are taken into account. Examples for such forecasting methods are regressive methods like the popular auto-regressive integrated moving average (ARIMA) models, artificial intelligence techniques like k-nearest neighbors or artificial neural networks (ANNs) as well as others like remote sensing models [113].

The accuracy of renewable energy forecasts generally increases when near real-time meter data and detailed static data (e.g., location, hardware information, panel orientation, etc.) are available for all interconnected systems [114]. As we want to maximize the utilization of renewable energy in our system and an adaptive reaction to changes is possible through the use of a descheduler, the longest forecast considered is day ahead.

Annotation of Nodes

In Kubernetes, annotations are used to attach arbitrary non-identifying metadata to objects[4], which in this case is utilized to make the external renewable energy data available to the scheduler and descheduler. Depending on the system attached and the degree of control over the connected hardware, this data can be more or less granular. Along with the interval timestamp and the nodes current consumption, the real-time and forecast renewable energy

4 https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/, *accessed 2021-08-21*

values are combined into a string and sent through kube-apiserver via the *patch_node(node, json)* function as key-value pair JSON payload to each node in the cluster.

Listing 4.1: Energy Data Node Annotation Function

```
def update_annotation(node_name, ts, consumption, renewables):

    # annotation body
    annotations = {
                "metadata": {
                    "annotations": {
                        "timestamp": ts,
                        "consumption": consumption,
                        "renewables": renewables
                    }
                }
            }

    # send to k8s
    response = k8s_api.patch_node(node_name, annotations)
```

4.3 Renewable-Aware Scheduler

The goal of the *green-k8s-scheduler* is to influence the scheduling decision in favor of nodes that can offer a higher availability of renewable energy at their location. Therefore, the scoring stage should be extended with custom logic that considers the data that is made available at the nodes by the *green-k8s-energy-management*. Currently there are four methods[5] to intervene in the scheduling process or to adapt it to individual needs. Each method has its advantages and disadvantages, which will now be discussed briefly.

4.3.1 Scheduling Extension Variants

Analysis of available Methods

1. The Ad-hoc variant[6]: In this variant, the standard kube-scheduler source code is modified directly in its core and recompiled to run as a single component. An own customization of the Kubernetes source code is generally not recommendend, independently from the affected component, as it severely limits compatibility and the possibility of seamless integration of later updates. This in turn also has an impact on the maintainability of the implementation, which decreases significantly. On the other hand such close integration has the advantage of

5 https://kubernetes.io/docs/concepts/extend-kubernetes/, *accessed 2021-08-21*

6 https://static.sched.com/hosted_files/kccncna20/49/kubecon-scheduler-enhancements_final.pdf, *accessed 2021-08-25*

a rather high performance and the absence of conflicts between scheduling decisions.

2. Scheduler Extender[7]*:* In contrast to the ad-hoc variant, a scheduler extender is implemented as independent component, which communicates with the kube-scheduler via webhook. The high degree of independence has the advantage of a very good compatibility and maintainability, for both the Kubernetes cluster and the extender. For each kube-scheduler, multiple scheduler extenders can be configured in the kube-scheduler configuration YAML along with default scheduling policies. Scheduler extenders can complement the standard scheduler in three stages: (Post)Filter, (Post)Score and Bind, meaning it does not replace the default scheduling decisions, but enhances them. A disadvantage can be a decreased performance due to the communication overhead between the two or more components.

3. Multiple Schedulers[8]*:* Another possibility is the deployment of your own scheduler(s) instead of or parallel to the default kube-scheduler. This can be useful if the workloads running on a cluster have very heterogeneous requirements and the schedulers needs to take care of different metrics. Each pod can then be assigned its own scheduler. However, multiple schedulers also entail the risk of conflicting scheduling decisions.

4. Scheduling Framework[9]*:* A fourth variant is to utilize the scheduling framework of Kubernetes, which allows to intervene in the scheduling process at all sub-stages. The scheduling framework allows to write so called plugins that can be compiled into the default kube-scheduler, while still be logically separated from the core, which combines a good performance with a good maintainability. Each plugin can extend one of the nine sub-stages, which makes the scheduler framework a good choice if fine grained extensions are needed.

Method Selection and Setup

Out of the four scheduler extension possibilities introduced before, the extender was chosen as it is easy to maintain and the implementation and deployment is a straightforward process, as other works already prove its applicability and can serve as a reference. Furthermore, it covers the requirement to be able to enhance the Scoring process. The development of a scheduler plugin was also considered as due to its recent introduction the Scheduler framework might be favorable in the future. However, at the beginning of the research for this thesis not all

7 https://github.com/kubernetes/community/blob/master/contributors/design-proposals/scheduling/scheduler_extender.md, *accessed 2021-08-25*
8 https://kubernetes.io/docs/tasks/extend-kubernetes/configure-multiple-schedulers/, *accessed 2021-08-25*
9 https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/, *accessed 2021-08-25*

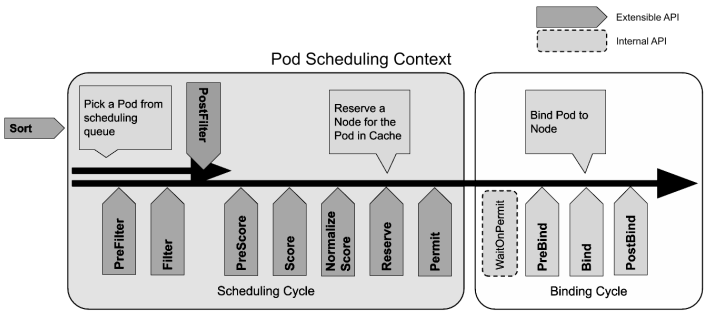


Fig. 4.5: Possible Extension Points for Plugins in the Scheduling Cycle

scheduler framework features were fully developed and only a few reference sources were available, as it just has recently been transferred to the stable version. The annotation based approach is also agnostic to the extension method, thus it can be universally applied if needed.

To create a scheduler extender there are two major steps to be done:

1. Configure the kube-scheduler to communicate with the extender container
2. Develop the scheduler extender image that implements the custom scheduling logic and exposes an HTTP server

The HTTP server needs to be accessible by the default scheduler as it will pass information about the Pod that needs to be scheduled, as well as a list of available nodes in the cluster that passed the filtering phase to the extenders */prioritize* endpoint for each Pod that needs to be scheduled. The extender will process this information and return a *HostPriorityList*, each with a calculated score per node. The default scheduler will then take this score into consideration when making the final decision on where to place the Pod.

4.3.2 Scheduler Configuration

To deploy the extender image successfully and enable it to communicate with the scheduler and the control plane, a composition of several configuration objects are needed, which are defined in the *scheduler.yaml* file:

- The scheduler policy ConfigMap like shown in Listing 4.2 tells the scheduler which predicates and priorities to use, how to order the predicates and how to weight the priorities. Furthermore, it gives the scheduler the list of extender URLs including the path parameter for the extension points in use (e.g. prioritizeVerb, filterVerb), of which can be multiple defined. As the scheduler and the extender container run in the same

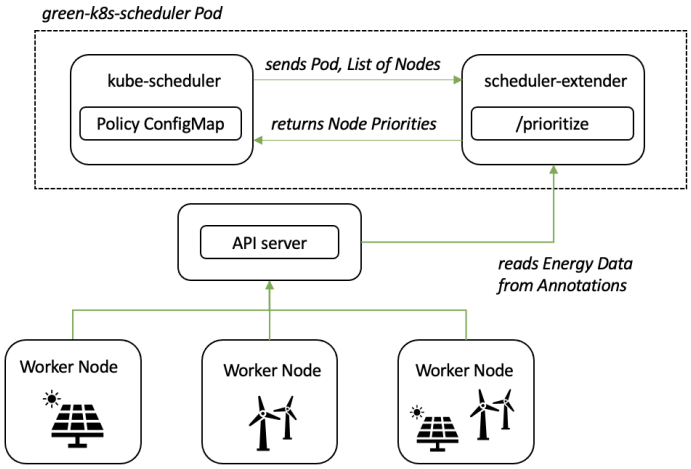


Fig. 4.6: Scheduler Extender Interaction

Pod, they can communicate via localhost. The policy is given in JSON format embedded into the ConfigMap.

Listing 4.2: kube-scheduler ConfigMap with custom Extender

```
{
    "kind" : "Policy",
    "apiVersion" : "v1",
    "predicates" : [
      {"name" : "PodFitsHostPorts"},
      {"name" : "PodFitsResources"},
    ],
    "priorities" : [
      {"name" : "LeastRequestedPriority", "weight" : 1},
      {"name" : "BalancedResourceAllocation", "weight" : 1},
    ],
    "extenders" : [{
      "urlPrefix": "http://localhost:80",
      "prioritizeVerb": "prioritize",
      "weight": 1,
      "enableHttps": false,
      "nodeCacheCapable": false
    }],
    "hardPodAffinitySymmetricWeight" : 10
 }
```

- The scheduler ConfigMap contains a KubeSchedulerConfiguration object that points to the Policy used by the scheduler and instructs the scheduler about its own name.

- A ServiceAccount object that provides an identity for the processes that run in the green-k8s-scheduler Pod to be able to authenticate against the kube-apiserver.

- A ClusterRoleBinding object that grants cluster-wide admin access to the ServiceAccount through role-based access control (RBAC), a method of regulating access to computer or network resources.
- A Deployment that deploys two containers in the green-k8s-scheduler Pod: An instance of the kube-scheduler that will connect to the extender, as well as the extender application container itself.

4.3.3 Custom Scoring Algorithm

The extender image implements its own custom logic to score the nodes that passed the filter phase, according to their available renewable energy excess shares. To be flexible in terms of different application scenarios for the scheduler, the configuration offers the possibility to apply five different modes to be able to be more flexible in terms of covering different types of applications to be scheduled. The configuration can be customized in the *scheduler.yaml* file and is applied at start time of the extender, by adding the desired value as environment variable to the container.

Total Score Calculation

The goal of the scoring process is to find the perfect fit in the remaining nodes, represented by a normalized score between 0 (worst) and 10 (best). Therefore, a multi-dimensional scoring model was developed, which consists of five steps.

1. **Calculate Renewable Excess:** The annotated data of each node is retrieved by the extender and parsed into a data structure. Depending on the mode, only relevant renewable energy data is processed further. Then, the current energy consumption per node needs to be calculated or retrieved from the annotations / energy management system, since we do not only want to look at the total availability of renewable energy at individual nodes, but utilize the relative available share. This is important for cases where the current consumption already exceeds the available renewable energy, as this particular node should not be preferred in the scoring, even if it has the highest total share of renewables. The current consumption is assumed to be steady over time as the runtime of current workloads is unknown to the scheduler, which is why it is subtracted from all renewable shares to calculate the remaining excess energy. As it is recalculated for each scheduling cycle, it always adapts to current given circumstances.
2. **Score per Value:** After the excess energy is calculated for each period, the first scoring phase starts, whereas an individual score is calculated for each renewable energy share

period per node like shown in Algorithm 1. In order to get a set of normalized scores per share, several iterations have to be performed. Therefore, all shares at the same index (e.g. at the index for the 1 hour forecast) at each node are compared. Once the highest and lowest values of renewables at a certain index has been found, the individual score can be calculated in relation to them. The scores per period and share are needed to avoid later imbalances in the total scoring due to different availability of renewables per period.

3. **Apply Weights:** The weights are applied to each scored period per node. The weighting optimizes scheduling towards the maximum runtime of each mode.

4. **Sum Up:** All weighted scores are added up. The sum of those weighted individual scores represents the basis for the total score of each node.

5. **Calculate Final Score:** The given sum per node goes through another normalization phase and results in the total score per node. This result is returned to the kube-scheduler and considered among the scores returned by the other priority functions.

Algorithm 1 green-k8s-scheduler Score Calculation Step 2

```
1:  normalizedScores := make(map[string][]float64)
2:  for currentShare := 0; currentShare < windowSize; currentShare++ do
3:      high := 1.0
4:      low := 0.0
5:      for node in nodeShares do
6:          high = math.Max(high, nodeShares[node][currentShare])
7:          low = math.Min(low, nodeShares[node][currentShare])
8:      end for
9:      for node, shares in nodeShares do
10:         normalized = ((shares[currentShare] - low) / (high - low)) * MAX_SCORE
11:         normalizedScores[node] = append(normalizedScores[node], normalized)
12:     end for
13: end for
14: return normalizedScores
```

Scheduling Modes

In general, the green-k8s-scheduler can be used for any kind of Job or workload that can be deployed in a Kubernetes cluster. However, the dependency on the availability of renewable energy as a decison-making factor for scheduling introduces a temporal dimension regarding

the runtime of workloads that has to be considered accordingly. For that matter, applications can basically be categorized into three types, according to their runtime [115]: Short-running, long-running and continuously running workloads, whereas short-running (batch) jobs, with a runtime of a few minutes represent the largest share of workloads executed in data centers. With the configuration of different modes, application developers and DevOps engineers are given the possibility to pick the most precise option for their respective type of workloads. The naming of the modes is carried out according a method called "T-Shirt sizes", known from Scrum, in order to increase user-friendliness.

Mode	Expected Application Runtime	Window / Renewable Data Considered
XS	up to 10 Minutes	now
S	up to 1 Hour	now - 1 Hour Forecast
M	up to 4 Hours	now - 4 Hours Forecast
L	up to 12 Hours	now - 12 Hours Forecast
XL	up to ∞ Hours	now - 24 Hours Forecast

Tab. 4.1: green-k8s-scheduler Scheduling Modes

While each node provides a total of 25 annotations, only the respective renewable energy data needed for the runtime that is relevant in the chosen mode are considered in the scheduling decision. The window of the data includes spans from the time of scheduling (now) up to the respective forecast depending on the chosen mode, defined in Table 4.1. By choosing the "S" mode for example, only the "real-time" 10-minute energy and the one hour forecast values are taken into account, while the "XL" mode takes all 25 annotated values into consideration.

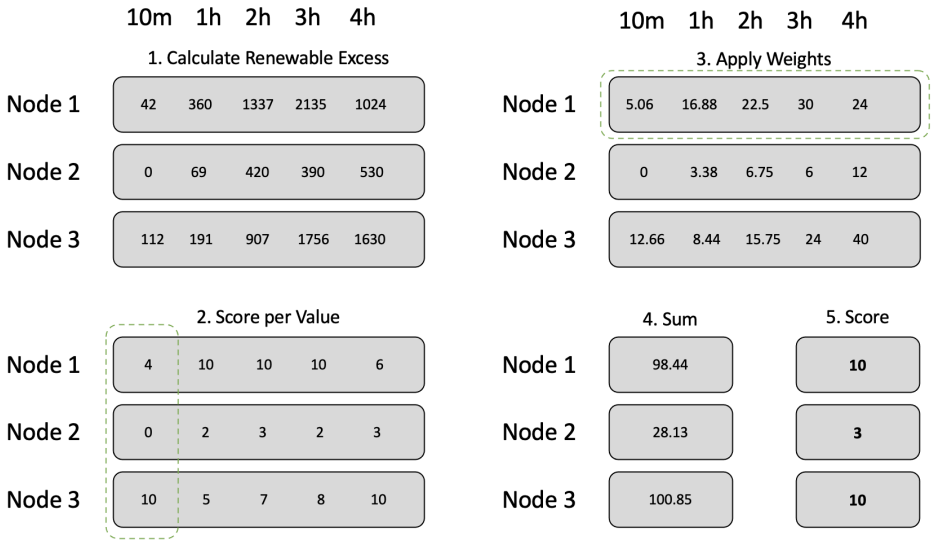


Fig. 4.7: Example Scoring of three Nodes with Mode "M" Scheduling

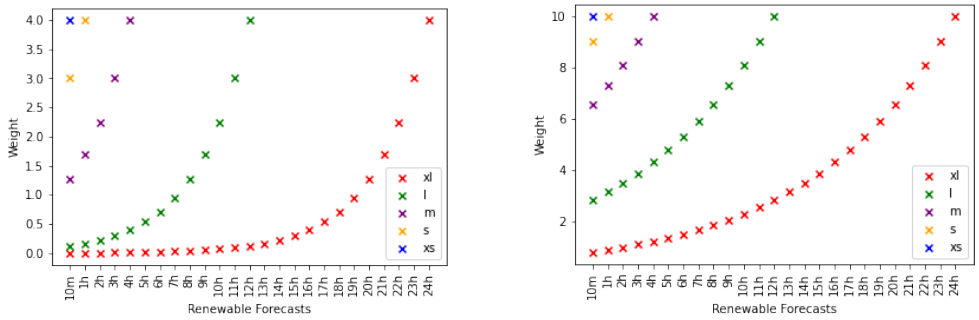
As we can see in the example with random numbers in Figure 4.7, in the end there could also be two nodes with the same score, even if they have slightly different total sums and therefore also different intermediate scoring results. In this case, the intermediate results for Node 3 would be 10, while Node 1 would have a result of around 9.7. Since the HostPriorityList, which is returned to the kube-scheduler, demands scores of type integer, the logic of the normalization function rounds the result of the score calculation to the nearest integer. However, the total score returned by the green-k8s-scheduler algorithm is only one of many priorities that can be considered by the kube-scheduler. The final scoring is somewhat nontransparent, as the scores returned by the other priority functions defined in the scheduling policy are not known to the outside, thus the final scheduling decision is dependent on the respective outcomes of these as well.

Weight Distribution

In order to obtain an appropriate score in each mode to select the best node with the highest average renewable energy over the whole application runtime, a proper weighting is needed which favors the relevant periods in the given data. For the calculation of the weight factors for each renewable value, an exponential distribution with geometric progression is used to apply a greater weight to the time windows in which the scheduled Pods are about to run.

$$weight_i = \frac{e^i}{1 - e} \tag{4.1}$$

The weight is calculated per value i in the chosen window according to the mode, whereas $i = \{0, 1, 2, ..., \text{number of renewable energy values} - 1\}$. The flexibility of this function allows for dynamic further integration of renewable values in the annotations (e.g. more forecasts, modes of finer granularity) if needed. For e holds $0 < e < 1$, with a default value of $e = 0.75$. If needed, the behavior of the weighting function can be changed with the adjustment of e like shown in Figure 4.8, e.g. to further optimize scoring in addition to the general usage of modes, which can be valuable especially if windows are rather wide like in the "XL" mode.



(a) e = 0.75 **(b)** e = 0.9

Fig. 4.8: Weighting of Renewable Forecast Values for different Modes

4.4 Renewable-Aware Descheduler

In addition to the green-k8s-scheduler, which is preferable for workloads that should not be interrupted, the *green-k8s-descheduler* can be used to further optimize the overall Pod distribution across node locations. By regularly checking the current renewable energy excess availability, it is able to react to short-term fluctuations and consequently evict Pods from nodes that offer worse conditions than others. Thanks to Kubernetes aspiration to restore the cluster state configured in the Deployment, the underlying system takes care of spinning up new Pods in the event of an eviction. The newly created Pod will automatically be scheduled by the green-k8s-scheduler as configured, thus be placed on the node with better conditions in terms of renewable energy availability.

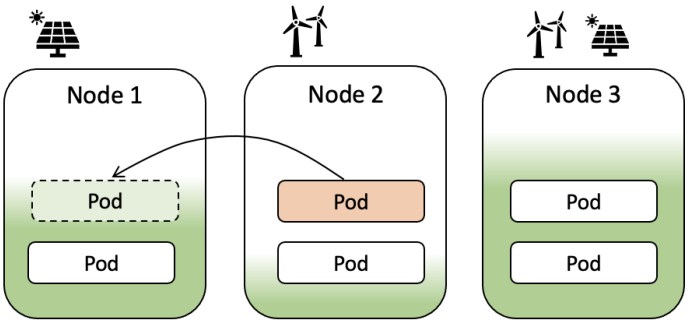


Fig. 4.9: Concept of Renewable-Aware Descheduling

As a kube-system Pod running on the master node, the green-k8s-descheduler is able to retrieve the current state of the cluster and its data respectively. Just as the scheduler, it reads the energy data from the node annotations, as introduced before. Having current node consumption and renewable energy production data at hand, the green-k8s-descheduler validates if the Pods are optimally distributed according to the availability of renewables.

However, an eviction is only triggered if there is an imbalance across nodes, thus an eviction is neither initiated if all nodes provide renewable excess energy nor if no node provides renewable excess energy. If there is an imbalance, the nodes with the highest and lowest renewable energy excesses are identified. Then, the descheduler checks how many Pods at the node with the lowest excess are eligible to be evicted and rescheduled. Additionally, a threshold factor can be configured in the *descheduler.yaml* as a "safety net" to reduce the likeliness of Pod redistribution, while ensuring the full coverage of the respective workload. Therefore, the total energy consumption and the number of Pods to be potentially evicted must be known. While iterating over the number of Pods on the node with the lowest excess, following condition has to hold true to start the eviction process:

$$highestExcess - consumptionPerPod * numberOfPods > lowestExcess * threshold \quad (4.2)$$

In this exemplary case in Figure 4.9, Node 2 cannot cover its two Pods consumption with renewable energy and thus has a negative excess (lowest excess), whereas Node 1 (highest excess) and Node 3 still have capacities. The task of the descheduler is to initiate the eviction of the Pod at Node 2, here shown in red. As Kubernetes strives towards its state of running five Pod replicas, it will initiate a Pod scheduling with the green-k8s-scheduler, which selects Node 1 for the replacement Pod (shown in green), as it offers the highest renewable excess energy. In combination with the green-k8s-descheduler, the green-k8s-scheduler should be configured to use either mode "XS" or mode "S", to fully utilize the short-term sensitivity and avoid unexpected behavior.

5 Evaluation

5.1 Experimental Setup

In the following chapters, the overall test setup is introduced, whereas the focus is set on the description of the underlying infrastructure as well as the simulation of the renewable data written into the annotations and the energy consumption of each node.

5.1.1 Kubernetes Cluster

The test environment for the evaluation of the green-k8s-scheduler and green-k8s-descheduler consists of a Kubernetes v1.22 cluster on EC2 virtual machines of the cloud provider Amazon Web Services. A managed Kubernetes service, such as Amazon EKS, was deliberately not used, since the access rights for configuring the master node components are restricted there. The setup of the cluster is carried out by the command line tool kOps[1] (Kubernetes Operations) version 1.21.0, which automatically provisions the required cloud infrastructure and its underlying networking configuration respectively. Therefore, a kOps user has to be created and equipped with different AWS IAM policies to gain access to the virtual resources. The number and size of the nodes is easily scalable, which makes it possible to perform the tests under different conditions, if needed. The cluster state is stored in an AWS S3 bucket. The green-k8s-energy-management will run as component external to the cluster on a MacBook Pro 3,1 GHz Dual-Core i5 with 8 GB RAM.

For the following test scenarios, a Kubernetes cluster consisting of one master node and four worker nodes of size t2.small is used, unless otherwise stated. A geo-distribution aspect to replicate a scenario with a cluster spanned over multiple locations equipped with different renewable energy equipment is considered with the use of four different AWS availability zones (us-east-1a, us-east-1b, us-east-1c and us-east-1d). The number of nodes could be higher in Fog or Edge computing scenarios, but this would not affect the functionality of the green-k8s-energy-management, green-k8s-scheduler or green-k8s-descheduler per se. The selected

1 https://github.com/kubernetes/kops, *accessed 2021-11-19*

configuration should therefore be sufficient to demonstrate the advantages of renewable aware scheduling as it offers four worker nodes that are equipped with four different renewable energy setups respectively. Besides the green-k8s-scheduler and descheduler Pods, the Kubernetes metrics-server is deployed in the cluster to provide detailed node and Pod metrics. For the test Pods to be deployed in the experiments, a custom container image (green-k8s-test[2]) is used, which contains a Python script that generates CPU load by doing looped factorial calculations.

5.1.2 Energy Management System

In the experiments, time and day when each test is carried out are chosen based on suitable renewable energy data availability that matches the requirements to demonstrate the scheduling and descheduling functionalities properly. First and foremost, several assumptions have to be made to be able to implement the given complex multi component system on AWS EC2 nodes without access to renewable energy hardware and neither cluster nor node level electricity consumption data. To be able to calculate the current available (excess) renewable energy at each node, two parameters have to be known to the system: The current input power / utilization of the system and the renewable energy output available at the nodes. For the tests, these values are simulated like described in the following paragraphs.

Current Energy Consumption / System Utilization

In the given test setup, the current energy consumption at each node is unkown, thus only assumptions can be made as there is to my knowledge no possibility to get any information about technical details of EC2 nodes, like the rated power of a system or the current energy consumption out of AWS. For the maximum rated power, a value of 10 kW per node is chosen, to match the approximate peak output of the renewable energy appliances at full utilization. The current energy consumption is calculated in the green-k8s-scheduler at the time of scheduling based on the nodes current CPU utilization data, gathered from the Kubernetes metrics-server. Research shows that CPU utilization and energy consumption correlate at least to some extend [116, 117], which makes it the best indication available under the given circumstances. A node with a CPU utilization of 42% would then have a resulting energy consumption of 4200 Watt (0.42 * 10 kW). In real environments, however, the corresponding values can easily be integrated.

2 https://hub.docker.com/r/timokraus/green-k8s-test, *accessed 2021-11-19*

Renewable Energy Output

The green-k8s-energy-management will generate data to simulate locally equipped renewable energy plants, such as solar panels or wind turbines as exemplified in Figure 5.1. The Python script equips nodes dynamically with either solar, wind or a mixed renewable energy equipment. A comparability with real world scenarios should be established, which is why the data for the annotations is generated based on measurements at a granularity of ten minutes from a station of the German Weather Service[3]. Solar and wind data are taken from the same weather station and time period respectively. In this particular case, the station with the ID 05705, located in Würzburg, Germany and recent time periods from the year 2020 were chosen as a basis. In addition to the familiar measurements such as temperature, hours of sunshine or rainfall, the German Weather Service also makes other measurement data available to the public.

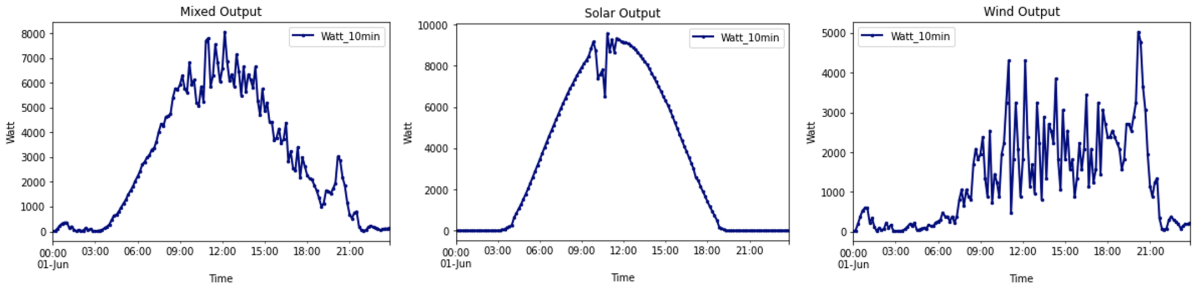


Fig. 5.1: Renewable Energy Production for different Equipments

For the calculation of the generated solar energy, the energy density values of the global radiation $Z_{radiation}$ as a **10-minute interval** sum in J/cm^2 are used. The following formula derived from Nguyen and Le [118] shows how to calculate the energy yield P_{solar} depending on the total area of photo-voltaic cells A_{panel} and its efficiency factor η_{panel}. Unknown influencing factors like the slope of the roof, azimuth, a temperature correction factor or other losses are ignored.

$$P_{solar} = A_{panel} \cdot Z_{radiation} \cdot \eta_{panel} \tag{5.1}$$

In order to emulate the solar panels of a smaller site or building, a panel surface area A_{panel} of $50 m^2$ is chosen, which is the approximate area needed[4] for a system with an installed power of 10 kW peak under standard testing conditions. This can be seen as a realistic value, as the

3 https://www.dwd.de/DE/leistungen/cdc/cdc_ueberblick-klimadaten.html, *accessed 2021-10-17*
4 https://echtsolar.de/kwp-pro-m2/, *accessed 2021-10-17*

average installed power in Germany for photo-voltaic systems on buildings was 7.5 kW peak in 2020 [119] and thus only slightly lower than the chosen reference. The efficiency η_{panel} strongly differs between modules and cell types. Under laboratory conditions, solar cells achieve an efficiency between 10% and 50% [120]. Under real conditions, however, an efficiency of 20% can be taken as realistic[5], which is why this is used as a reference value for the current calculations. Time and unit-related conversion factors are taken into account in the implementation.

The second renewable energy source to feed the renewable aware scheduler will be a small scale wind turbine. The German Weather Service also provides wind speed data in various granularity. Like for the solar radiation data, the **10-minute interval** is also chosen for the wind velocity. The following formula is derived from [121], ignoring the aerodynamic efficiency coefficient for simplification and as it is not know for the given turbine, to calculate the hourly energy yield P_{wind}:

$$P_{wind} = \frac{1}{2} \cdot \pi \cdot r^2 \cdot v^3 \cdot \rho_{air} \cdot \eta_{turbine} \tag{5.2}$$

When selecting the wind turbine parameters, care was taken to ensure a similar nominal power as the solar panels in order to create overlaps of the output curve with the solar system to challenge the (de-)scheduler, which is why the values used to calculate the output are based on a 10 kW peak model[6]. In case of high wind speeds in the data, the maximum output is also capped at this level to take technical limitations into account. As an alternative to the scenario with a single 10 kWp system, it is of course also conceivable to calculate with the accumulated energy production of many small wind turbines, which are for example better suited for inner-city applications depending on the node location. However, in our equation, the radius r equals 5.1 meters (based on the aforementioned model), the velocity v is taken from the weather data, air density ρ is given as an approximate value[7] of 1.2 kg/m^3 and the efficiency η is assumed to be 50% - a realistic value for modern wind turbines[8].

Renewable Energy Forecasts

Since in this work the generation data of the renewable energy sources are based on historical weather data, a simplified approach is chosen to generate the forecast values, which assumes

5 https://www.photovoltaik.org/wissen/photovoltaik-wirkungsgrad, *accessed 2021-10-17*
6 https://www.wind-turbine-models.com/turbines/1636-tuge-10-kw, *accessed 2021-10-17*
7 https://www.dwd.de/DE/service/lexikon/Functions/glossar.html?lv2=101518&lv3=607748, *accessed 2021-10-17*
8 https://www.wind-energie.de/themen/anlagentechnik/funktionsweise/, *accessed 2021-10-17*

perfect predictions. The forecasts are available as a average availability of renewable energy in periods of hourly granularity from the upcoming hour until the upcoming 24 hours. The implementation of forecasting models like ARIMA would go beyond the scope of this thesis. Instead, the predictions are formed by the mean of a rolling window of the calculated values for the 10-minute periods, which represents the average available power output over the given period of each forecast, respectively.

5.2 Experiments: Renewable-Aware Scheduling

For the evaluation of the green-k8s-scheduler, different scenarios will be introduced to show its general behavior and functionality. The scenarios will increase in complexity from basic functionality like the pure placement decision for a single Pod to a scheduling of multiple Pods. In the experiments, different scheduler configurations are used to evaluate the influence of the green-k8s-scheduler. The full configuration of the scheduling Policy is shown in Figure 5.2, whereas everything except the extender, here outlined in red, is is referred to as **default scheduler**. If we speak of the **green-k8s-scheduler**, the extender, here outlined in green, is also included in the scheduling logic.

```
{
"kind" : "Policy",
"apiVersion" : "v1",
"predicates" : [
  {"name" : "PodFitsHostPorts"},
  {"name" : "PodFitsResources"},
  {"name" : "NoDiskConflict"},
  {"name" : "MatchNodeSelector"},
  {"name" : "HostName"}
],
"priorities" : [
  {"name" : "LeastRequestedPriority", "weight" : 1},
  {"name" : "BalancedResourceAllocation", "weight" : 1},
  {"name" : "ServiceSpreadingPriority", "weight" : 1},
  {"name" : "ImageLocalityPriority", "weight" : 1},
  {"name" : "EqualPriority", "weight" : 1}
],
"extenders" : [{
  "urlPrefix": "http://localhost:80",
  "prioritizeVerb": "prioritize",
  "weight": 1,
  "enableHttps": false,
  "nodeCacheCapable": false
}],
"hardPodAffinitySymmetricWeight" : 10
}
```

Fig. 5.2: Default and green-k8s-scheduler Policy Configuration

To use the green-k8s-scheduler application in Kubernetes, it was build as a Docker Image and provided publicly on DockerHub[9]. To deploy the container, the *scheduler.yaml* manifest has to be applied to the cluster, using the *kubectl* command line tool. By checking the Pods

9 https://hub.docker.com/u/timokraus, *accessed 2021-10-14*

in the *kube-system* namespace with *kubectl get pods -n kube-system -o wide*, the green-k8s-scheduler can be spotted on the Master node next to other Control Plane components and is ready to be used. The experiments are carried out on a four node cluster, with each node having a unique renewable energy equipment like follows:

- Node 1: **Solar Equipment**
- Node 2: **Wind Equipment**
- Node 3: **Mixed Equipment** (60% Solar / 60% Wind)
- Node 4: **No Equipment**

The results are obtained from the scheduler (and extender) log outputs or dedicated evaluation scripts.

5.2.1 Single Pod Scheduling

The following tests show the scheduling decision for a single Pod for each scheduling mode of the green-k8s-scheduler (XS, S, M, L, XL) compared to the decision of the default scheduler. For this purpose, the final scores per node of both schedulers are compared and it is examined how the selected nodes differ on the basis of available renewable energy over the respective Pod runtime. For each mode, a Pod runtime according to the respective period covered by each mode is chosen. The tests are carried out on different (simulated) days and times to consider changing external influences. To provide an overview of general metrics and scoring results, a table is provided for each test. The "Ø Renewables" column shows the average available renewable energy over the given runtime of the respective test. The default scheduler scores (Score DS) and the green-k8s-scheduler score (Score GKS) are given to comprehend the Pod placement decision of the respective scheduler, highlighted by the bold score. The "Coverage" column shows the relative share of the workloads energy consumption which is covered by renewable energy at the respective node. Furthermore, two graphs resulting from each tests log data analysis are given. In the first graph, the available renewable energy at each node can be compared to the consumption of the test Pod. In the second graph, the potential coverage of the Pods consumption over the runtime by the renewable energy available at the nodes is presented. Additionally, the green area represents the utilization increase of renewables due to the usage of the green-k8s-scheduler in comparison to the default scheduling decision, while the red area indicates a lower utilization.

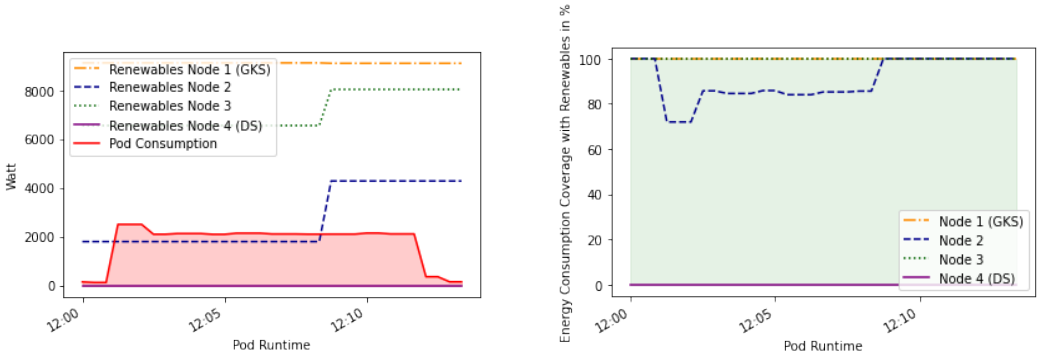
a) XS Mode - 10 Minutes Runtime - 01.06.2020 12:00 p.m.

In the first test, a Pod that runs for approximately 10 minutes is scheduled on the first of June 2020, at 12:00 p.m. using the XS mode of the green-k8s-scheduler and the default scheduler respectively. The resulting scores in Table 5.1 show that the default schedulers scoring results are almost evenly distributed across nodes. As each node provides more or less the same prerequisites, no special deviations occur during default scheduling and the Pod is placed randomly among the highest rated nodes.

Node	Equipment	Ø Renewables	Score DS	Score GKS	Coverage
1	Solar	9157.3 W	27	**37**	100.0 %
2	Wind	1816.2 W	26	28	90.7 %
3	Mixed	6584.1 W	27	34	100.0 %
4	None	0 W	**27**	27	0.0 %

Tab. 5.1: Scoring Comparison Test 5.3.1 a)

In this case, the default scheduler chose the worst possible node in terms of renewable energy availability (the one without any local energy appliances) and therefore causes a higher CO_2 footprint for the given workload than the node selected by the green-k8s-scheduler, which can cover 100% of the given demand. The additional scoring stage introduced by the green-k8s-scheduler also takes the results of the default scheduler into account, but additionally includes the energy dimension in the evaluation.



(a) Total Consumption and Production **(b)** Pod Consumption Coverage

Fig. 5.3: Energy Consumption vs. Renewables Production (a) and its relative Workload Coverage (b)

At any other node, the demand would have at least partly been covered like shown in Figure 5.3 (a) and (b) and Table 5.1. The given result is a first indication that the default schedulers randomness in terms of renewable awareness cannot compete with the green-k8s-scheduler,

which clearly chose the solar equipped node, providing the maximum renewable energy at noon. Furthermore, out of all nodes, the green-k8s-scheduler chose the optimum for the given Pod runtime. The green area in Figure 5.3 (b) represents the possible improvement in the utilization of the available renewable energies when using the green-k8s-scheduler. Due to the fact that Pods are distributed randomly in case a node receives the same score, the average coverage among those has to be calculated in order to establish comparability with the green-k8s-scheduler. Considering the nodes scored with a total of 27, the default scheduler would achieve an average coverage of 66.7 %, while the green-k8s-scheduler achieves a coverage of the given workload with renewable energy of 100.0 %, which is an increase of 49.9 %.

b) S Mode - 1 Hour Runtime - 01.06.2020 06:00 p.m.

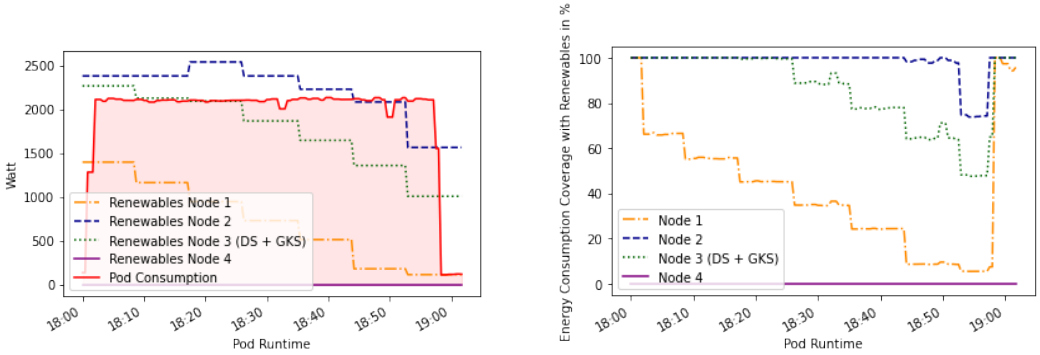
The next test case, where a Pod with a runtime of an hour is scheduled, similar findings as in the first test can be obtained. The default scheduler chose a random node out of three with the same highest rank, while the green-k8s-scheduler influences the scoring with regard to the decision in favor of the greenest node.

Node	Equipment	Ø Renewables	Score DS	Score GKS	Coverage
1	Solar	611.6 W	27	31	41.9 %
2	Wind	2201.4 W	26	36	97.9 %
3	Mixed	1687.8 W	**27**	**36**	86.0 %
4	None	0 W	27	27	0.0 %

Tab. 5.2: Scoring Comparison Test 5.3.1 b)

One notable finding resulting from this test is that the random selection of a node is also performed by the green-k8s-scheduler in this case, as the the slightly smaller initial score of the default scheduler at Node 2 (26) and the higher score at Node 3 (27) lead to a draw between nodes after the scoring of the green-k8s-scheduler which assigns the score of 26+10=36 to Node 2 and 27+9=36 to Node 3 like shown in Table 5.2.

The tight gap between the scores leads to a non-optimal scheduling decision in terms of renewable energy coverage and the same result for both schedulers, depicted in Figure 5.4. However, the randomness of the default scheduler in this matter cannot provide deterministic results, whereas the green-k8s-scheduler did at least give the highest score to the two best nodes in terms of renewable energy. To change the result more clearly in favor of the greenest node, a higher weighting of the extender in the scheduler configuration would be conceivable. Even though both schedulers picked the same node coincidentally in this case, one should consider the highest scored nodes as possible scheduling outcomes. This would result in



(a) Total Consumption and Production **(b)** Pod Consumption Coverage

Fig. 5.4: Energy Consumption vs. Renewables Production (a) and its relative Workload Coverage (b)

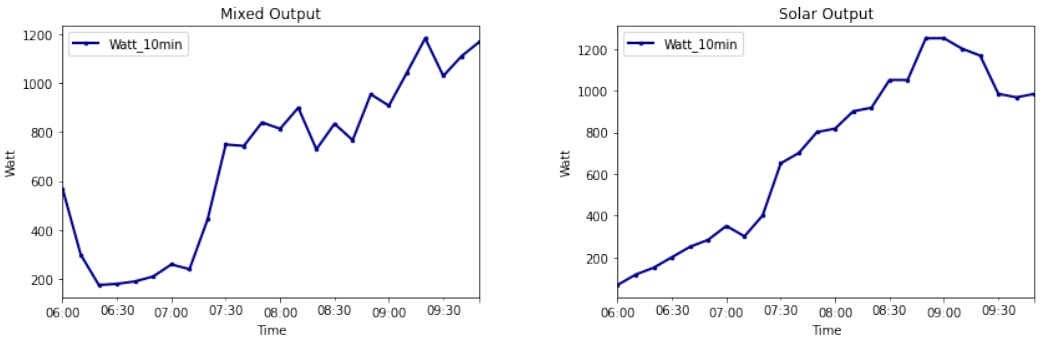
an average workload consumption coverage with renewable energy of 42.6 % by the default scheduler, while the green-k8s-scheduler would achieve an average coverage of 92.0 %. This is an increase of 116 %.

c) M Mode - 3 Hours Runtime - 14.10.2020 06:00 a.m.

In the third test, the default scheduler got lucky and chose the node with the supposedly highest renewables available (Node 1) comparing the values in Table 5.3, even if the scoring is not clearly pointing to any specific node here either. In contrast, the green-k8s-scheduler ranks the node with mixed equipment and slightly less renewables available in total highest (Node 3). This has basically two reasons: The first reason is that in mode M, the green-k8s-scheduler optimizes the scheduling decision for a runtime of four hours and the wind velocity got stronger between 09:00 and 10:00, resulting in a higher renewables production, shown in Figure 5.5. The second reason is that at the time scheduling, those two nodes had a slightly different base consumption (Node 1: 153.27 W, Node 3: 109.01 W), which is considered in the calculation for the available renewable excess energy at each node and can be obtained from the green-k8s-scheduler logs.

Node	Equipment	Ø Renewables	Score DS	Score GKS	Coverage
1	Solar	635.7 W	**27**	36	35.2 %
2	Wind	312.0 W	26	32	23.9 %
3	Mixed	568.6 W	27	**37**	33.7 %
4	None	0 W	27	27	0.0 %

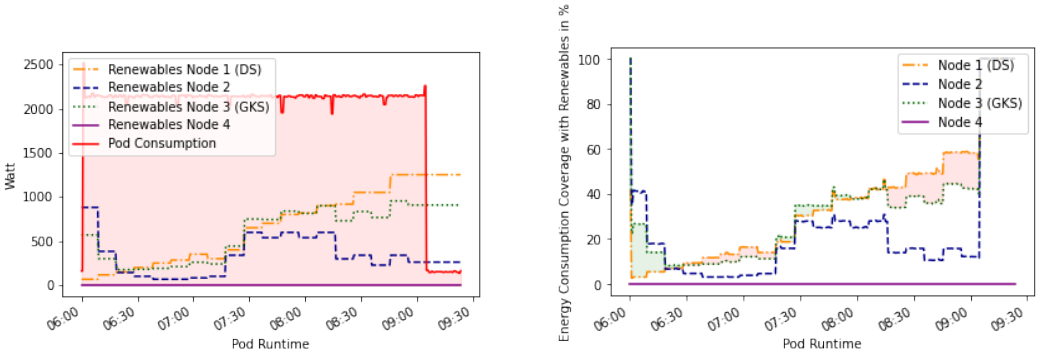
Tab. 5.3: Scoring Comparison Test 5.3.1 c)



(a) Output Mixed Equipment **(b)** Output Solar Equipment

Fig. 5.5: Renewable Energy Output for Mixed and Solar Equipment 14.10.2020 06:00 - 10:00 a.m.

The conditions observed resulted in a total availability of 607.69W / 4h for Node 3 and 583.43W / 4h for Node 1, which would have made Node 3 a better pick for the four hour period. However, after deduction of the base consumption, the total renewable excess energy available for the three hour period for Node 1 (482.43 W) and Node 3 (459.59 W) only differ slightly, thus the total "loss" is tolerable, considering the pick of the default scheduler was rather random. Furthermore, with an average coverage of 19.6 % among the highest scored nodes by the default scheduler, the green-k8s-scheduler still achieves an average increase in renewable energy coverage of 71.1 %.



(a) Total Consumption and Production **(b)** Pod Consumption Coverage

Fig. 5.6: Energy Consumption vs. Renewables Production (a) and its relative Workload Coverage (b)

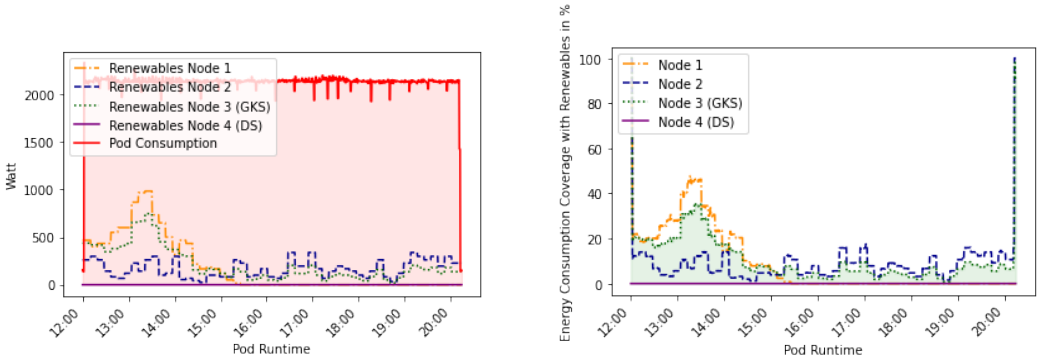
d) L Mode - 9 Hours Runtime - 25.12.2020 12:00 p.m.

In addition to the differences between the default scheduler and green-k8s-scheduler that have already been made clear in the previous tests, an interesting observation can be made in the test with a workload duration of 9 hours in mode L.

Node	Equipment	Ø Renewables	Score DS	Score GKS	Coverage
1	Solar	182.6 W	27	35	9.1 %
2	Wind	166.5 W	26	31	8.7 %
3	Mixed	209.5 W	27	**37**	10.7 %
4	None	0 W	**27**	27	0.0 %

Tab. 5.4: Scoring Comparison Test 5.3.1 d)

Like we can obtain from the graph in Figure 5.7, the the availability of renewables at Node 1 is clearly higher for the first three hours of the Pod runtime and at Node 2 it is higher for the remaining hours. However, thanks to the consideration of the corresponding forecasts in the L mode and a proper weighting, the green-k8s scheduler chooses the more favorable node with a mixed equipment, which provides a higher total utilization of renewables in the long run like shown in Table 5.4. For uninterruptible workloads, the Pod placement of the green-k8s-scheduler is again optimal, with an average increase in workload consumption coverage with renewables of 62.1 %.



(a) Total Consumption and Production

(b) Pod Consumption Coverage

Fig. 5.7: Energy Consumption vs. Renewables Production (a) and its relative Workload Coverage (b)

e) XL Mode - 48 Hours Runtime - 22.09.2020 06:00 a.m.

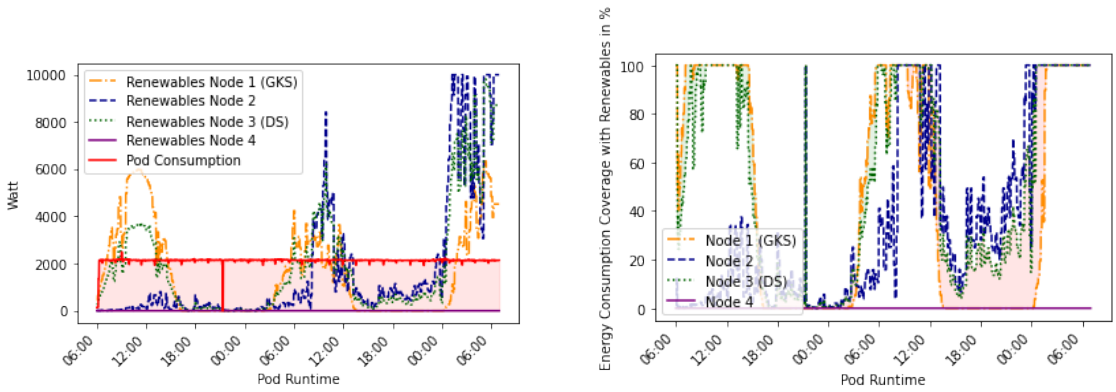
In the last test for the longest runtime of 48 hours scheduled in mode XL, the limits of the green-k8s-scheduler become clear. While Figure 5.8 suggests that the node where the Pod was placed by the green-k8s-scheduler (Node 1) is still the best choice for the first 24 hours, the circumstances change in the following period and the node chosen by the default scheduler would probably be the better choice (Node 3).

Also, like we can deduct from Table 5.5 the average availability of renewables over the 48 hour period is not very meaningful here, as the generation peaks which go beyond the

Node	Equipment	Ø Renewables	Score DS	Score GKS	Coverage
1	Solar	2096.1 W	27	**37**	46.2 %
2	Wind	783.9 W	26	27	36.5 %
3	Mixed	1243.6 W	**27**	34	52.2 %
4	None	0 W	27	27	0.0 %

Tab. 5.5: Scoring Comparison Test 5.3.1 e)

consumption of the Pod only suggest a higher availability, whereas the real coverage is higher even with a lower average renewable energy available when particularly long periods with high fluctuations are considered. However, the green-k8s-scheduler can only take a maximum of up to 24 hours ahead into account and thus cannot guarantee for sustainable Pod placements in any period beyond. If the Pod had been placed on one of the other nodes with a score of 27, the green-k8s-scheduler would have achieved at least the same or better performance. The coverage with renewables is in average thus still 59.3 % higher using the green-k8s-scheduler.



(a) Total Consumption and Production

(b) Pod Consumption Coverage

Fig. 5.8: Energy Consumption vs. Renewables Production (a) and its relative Workload Coverage (b)

To improve the placement of long and continuously running Pods, a descheduling mechanism should be considered, as the green-k8s-scheduler faces its limits at this point. Optionally, forecasts for longer periods can also be integrated if available to the system.

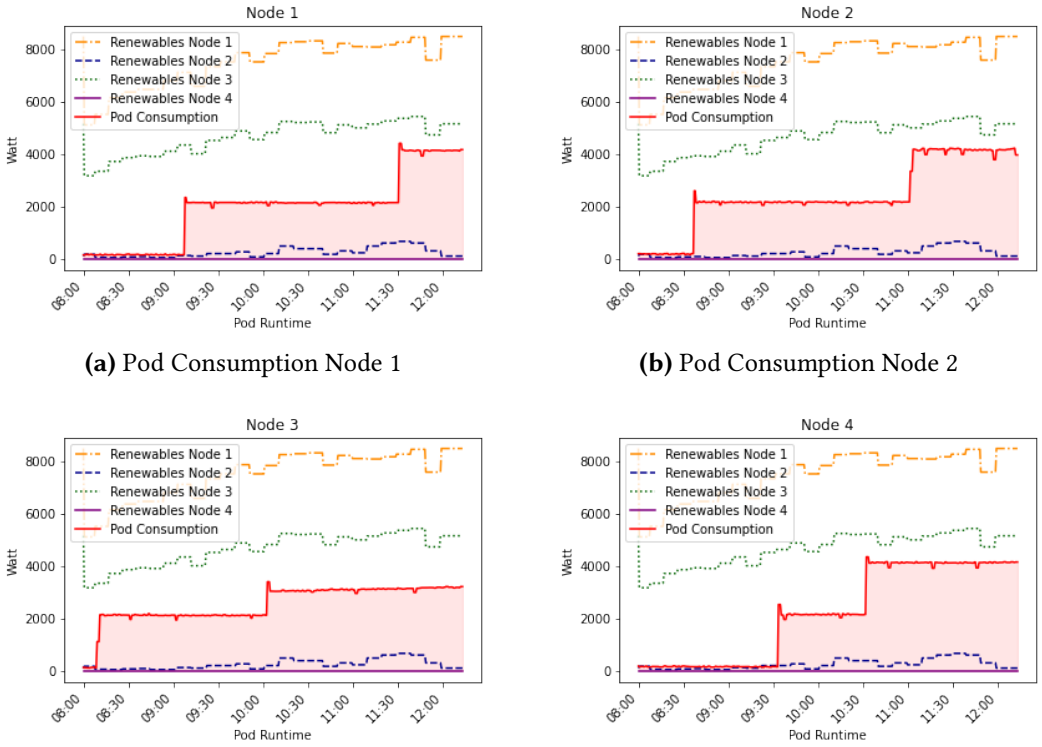
5.2.2 Multi Pod Scheduling

After the general behavior of the default scheduler and green-k8s-scheduler has been made clear in the previous tests, a further factor is now to be taken into account. Therefore, the following tests are carried out to evaluate the behaviour of the schedulers when handling multiple Pods.

a) 01.08.2020 08:00 a.m. - 12:00 p.m. - +1 Pod every 30 Minutes (up to 8) - Mode M

In the first test case, the Pod replicas are constantly increased over a period of four hours. Approximately every 30 Minutes, a new Pod is deployed into the cluster. Due to a constant increase in the number of Pods in the cluster by increasing the Pod replicas in the green-k8s-test Deployment, the existing workload is to be included in the scheduling decision. Figure 5.9 shows the behaviour of the default scheduler, while in Figure 5.10, the scheduling decisions of the green-k8s-scheduler are made clear. For the comprehensibility of scheduling decisions of the green-k8s-scheduler, Table 5.6 additionally lists all scores that were awarded to each node.

In Figure 5.9, we can see a uniform distribution of all Pods over the whole period. As the default scheduler considers mostly resources like memory and CPU utilization, the renewable energy availability is not taken into account at all. This leads to a waste of green energy, clearly visible in Graph (c) and (d), where half of the Pods are placed, even in the absence of renewable energy production at the node. In the end, each node runs two Pods in total. With the given distribution of the workload among the nodes, a coverage of the total consumption by renewable energy of 53.1% is achieved.



(a) Pod Consumption Node 1

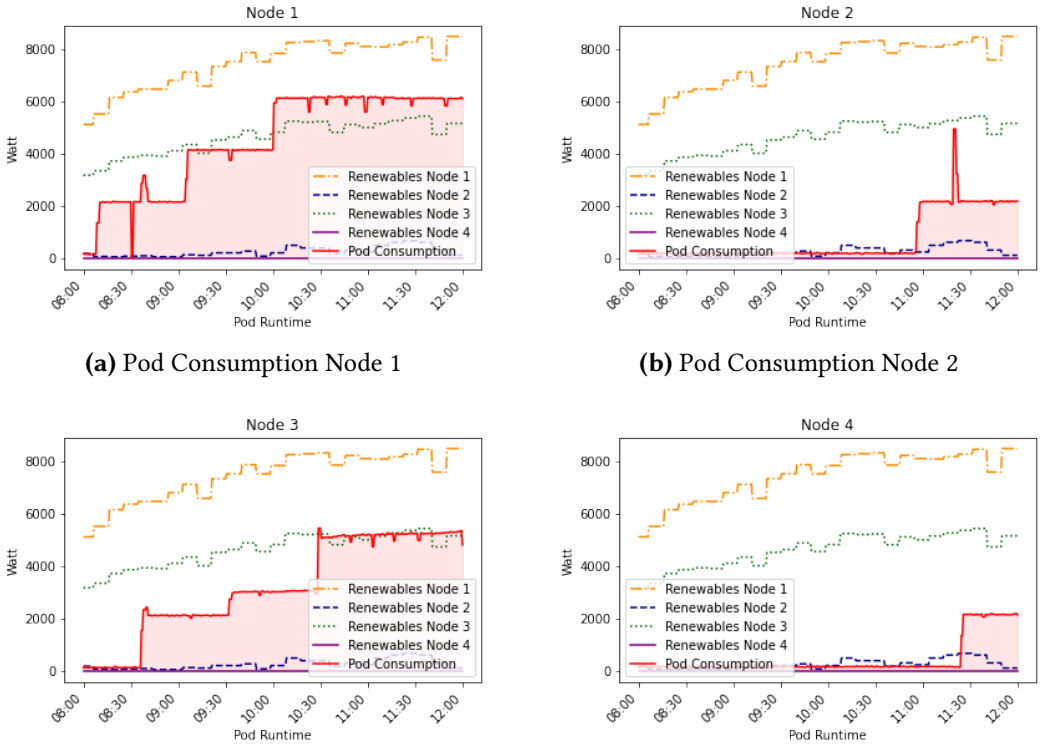
(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.9: Energy Consumption Production with increasing Workload - Default Scheduler

As soon as the green-k8s-scheduler is used for scheduling, we can clearly see a different distribution of Pods, shown in Figure 5.10. The first six Pods are placed on nodes with a high renewable energy availability (Node 1 and Node 3) which are able to cover the energy consumption of the workload deployed. While Node 1 would theoretically still have capacity to cover more Pods with renewables towards the end of the test period, the scores of the default priorities on balanced resource allocation have a higher impact due to the uneven distribution of the load, which results in a significantly lower total score, despite the highest score of 10, given by the green-k8s-scheduler extender. This can be obtained from Table 5.6, which gives more detailed insights in the resulting scores assigned to the nodes for each Pod. In contrast to the default scheduler, the green-k8s-scheduler achieves a coverage of the consumption by renewable energy of 91.0%, which represents an increase of 71.4%.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.10: Energy Consumption Production with increasing Workload - green-k8s-scheduler

One thing to be critically noticed is the scoring of Node 4 for the placement of Pod 9. Despite its non-existent renewable energy equipment, it receives a score of 9 by the green-k8s-scheduler. This score results from the normalization of scores over the whole range of available renewables - including negative values, which are caused by the consumption at Node 3 and Node 4. This mechanism is intended to favor nodes that are expected to offer

higher surpluses of renewables as load decreases (e.g. if Pods terminate), which cannot be the case here as there is no existing load on Node 4. However, with the given configuration, the green-k8s-scheduler significantly outperforms the default scheduler in terms of renewable energy utilization, thanks to the preference of Node 1 and Node 3. Furthermore, the given result can also meet high availability requirements if needed, as all nodes run at least one Pod of the Deployment.

Approx. Time	Pod	Score Node 1	Score Node 2	Score Node 3	Score Node 4
08:00:00	1	**37 (10)**	26 (0)	33 (6)	27 (0)
08:30:00	2	26 (10)	19 (0)	**29 (9)**	20 (0)
09:00:00	3	**24 (10)**	19 (0)	21 (5)	19 (0)
09:30:00	4	23 (10)	20 (1)	**25 (8)**	19 (0)
10:00:00	5	**23 (10)**	20 (1)	18 (5)	19 (0)
10:30:00	6	18 (7)	22 (3)	**24 (10)**	19 (0)
11:00:00	7	21 (10)	**26 (7)**	12 (0)	22 (3)
11:30:00	8	21 (10)	17 (1)	15 (3)	**28 (9)**

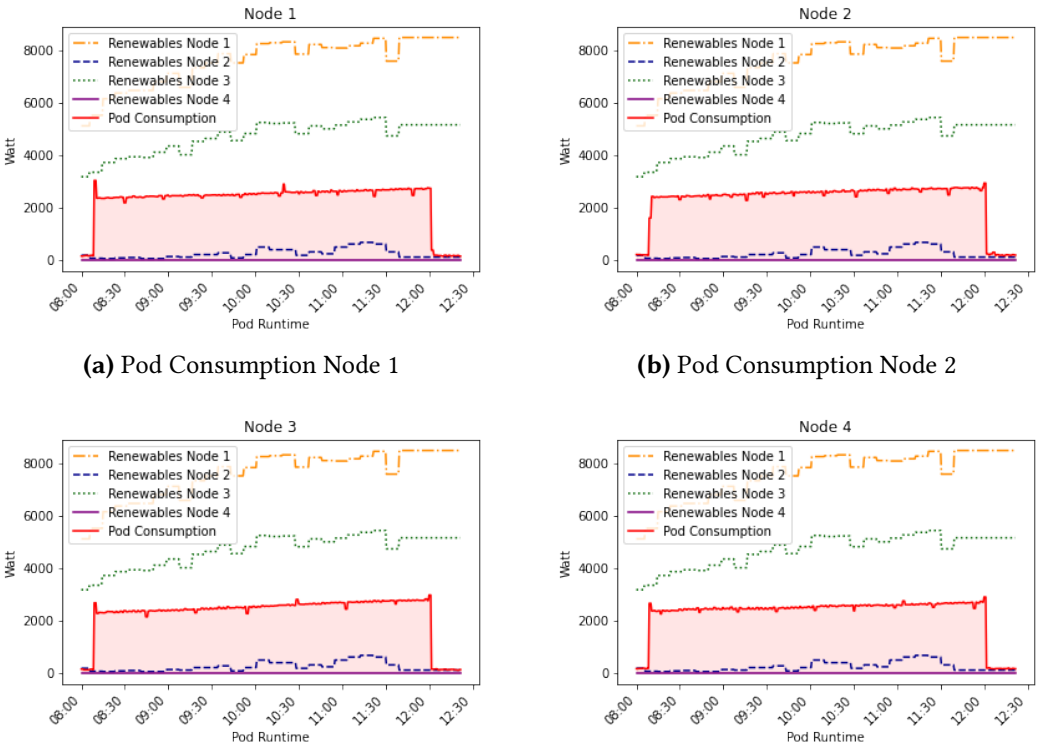
Tab. 5.6: Total Score and (GKS) Score for each Pod Placement by the green-k8s-scheduler

b) 01.08.2020 08:00 a.m. - Instant Deployment of 8 Pods - Mode M

In the second multi Pod scenario, the green-k8s-test Deployment with 8 replicas is instantly applied to the cluster to show the behaviour of the green-k8s-scheduler compared to the default scheduler with multiple Pod placements at once.

Figure 5.11 shows the behavior of the default scheduler. As expected, the Pods are distributed evenly over the available nodes. This result reflects the behavior of the default scheduler which could already be observed in the previous tests so far and therefore does not come as a surprise. As the Pods are deployed at once, the distribution mainly happens based on the ServiceSpreadingPriority, which assigns higher scores to nodes that are not yet running a Pod. Important to know, the BalancedResourceAllocation priority assigns almost the same score to each Pod as the scheduling and deployment happens in a few seconds at most, whereas the CPU and Memory consumption of the nodes is not yet measurably affected.

The fact that CPU and Memory have not yet undergone any noticeable changes in the short time of deployment is also evident when scheduling with the green-k8s-scheduler. As the scoring relies also on consumption data that is calculated based on CPU utilization, this factor cannot be taken into account, just like in real scenarios the energy consumption data would probably not be available instantly.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

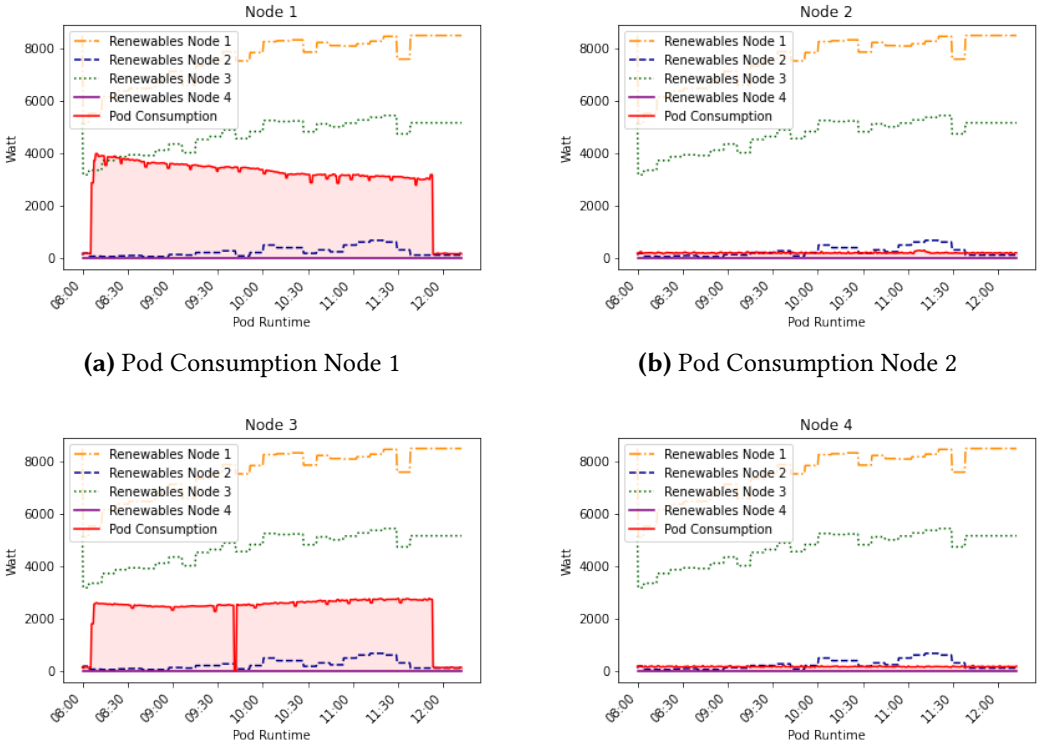
(d) Pod Consumption Node 4

Fig. 5.11: Energy Consumption Production with instant Workload - Default Scheduler

Analysing the log outputs shows that for each single Pod, Node 1 gets the highest score (10), followed by Node 3 (6), Node 2 (0) and Node 4 (0). As changes to the nodes consumption cannot be taken into account so quickly, this might lead to inadequate scheduling decisions in other scenarios. However, in symbiosis with the default priorities the Pods get scheduled properly, as the ServiceSpreadingPriority had enough weight to spread the Pods at least to two Nodes (1 and 3), with five and three Pods placed respectively, depicted in Figure 5.12. In this particular case, it results in a higher total utilization of renewables than in Test 5.2.2, as the nodes with a low availability of renewables are not considered at all. Without the influence of the ServiceSpreadingPriority, the number of Pods that get placed on Node 1 would only be limited by the Pods resource request limits defined in the deployment.yaml, which in this case is 150m CPU or 15% of one core. On the given single core machine, a total of six Pods could have been placed at Node 1 before reaching the resource limit. Afterwards, Node 1 would not pass the filter phase and the remaining two Pods would be placed at Node 3.

In conclusion, the green-k8s-scheduler outperforms the default scheduler with a renewable energy coverage of the total consumption at all nodes of 96.4% versus 53.6%, which is an

increase of 79.9% (100.0% cannot be achieved as the base workload at Node 2 and Node 4 are also taken into account). Furthermore, comparing Figure 5.12 and Figure 5.11 a concentration of workload to less nodes seems to have an additional positive effect on the total consumption, which does not increase linear with the number of Pods deployed. This could be seen as a beneficial side effect of the green-k8s-scheduler as well, if service distribution is less important.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

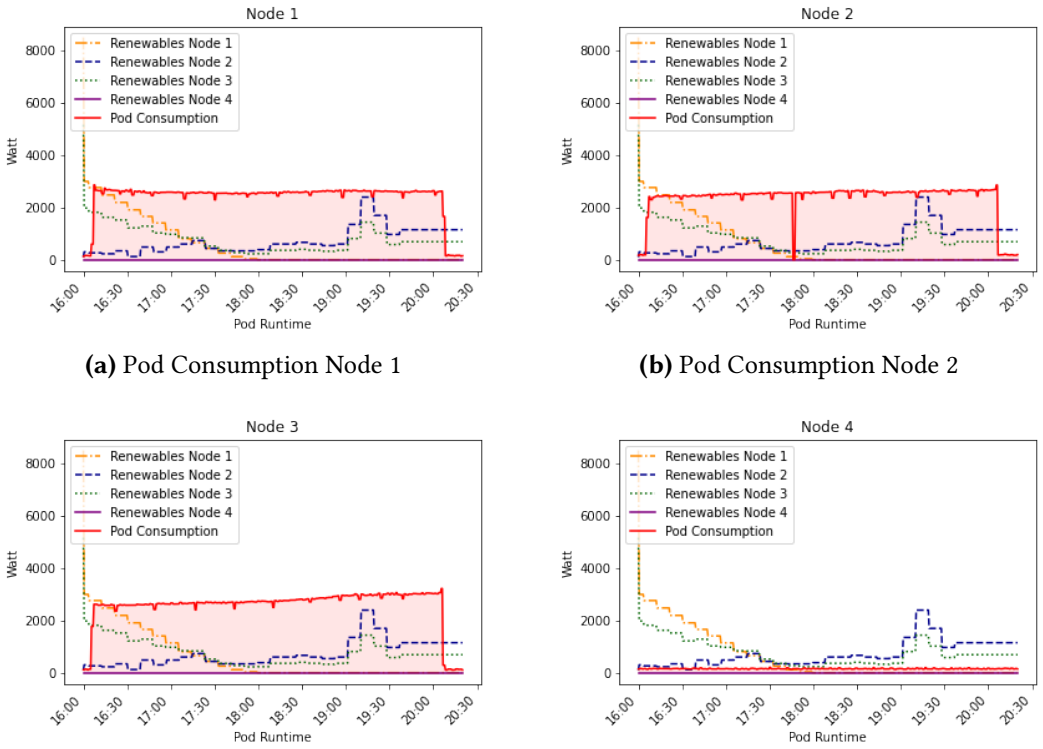
(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.12: Energy Consumption and Production with instant Workload - green-k8s-scheduler

c) 04.04.2020 04:00 p.m. - Instant Deployment of 8 Pods - Mode M

The following test case examines another instant multi Pod deployment with less predominant renewable generation distributions on two nodes, but a more even distribution to see if the green-k8s scheduler scoring misses the opportunity to consider all available renewables to cover the Pods consumption, even if they turn out to be significantly lower. Recapitulating the previous test, one would expect a distribution of Pods to two nodes with the highest scores. The graphical analysis of the default scheduler decisions is omitted in this case, as it would not provide further insights which are not known yet.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.13: Energy Consumption and Production with instant Workload - green-k8s-scheduler

Looking at the results of the test in Figure 5.13, one can say that the the expectations were exceeded. If the different nodes provide a rather similar renewable energy excess over the test period, the Pods are obviously scheduled to the correct nodes even during instant deployment. Analysing the logs, one can see that the equal scores of Node 1 (10) and Node 3 (10) given by the green-k8s-scheduler extender for a similar availability of renewables, lead to the consideration of three nodes. Although Node 2, with a score of 6 only gets two Pods assigned, while Node 1 and 3 are assigned three pods each, an optimal utilization of renewables is still achieved, with a total coverage of 29.0% for the given workload, which is an increase of 26.6% compared to the coverage with the default scheduler (22.9%). This shows that the green-k8s-scheduler achieves very good results in combination with standard policies.

5.3 Experiments: Renewable-Aware Descheduling

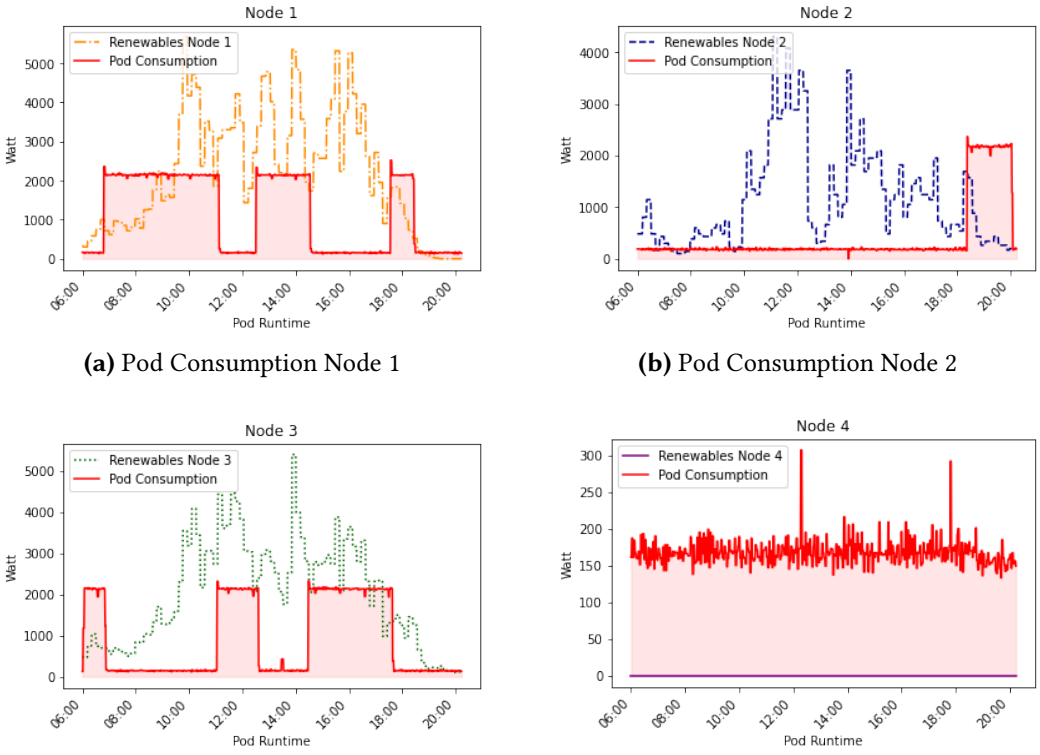
After extensive evaluation and analysis of the green-k8s-scheduler, a further improvement in renewable energy utilization shall now be examined with the usage of the green-k8s-descheduler. As we want to achieve an adaptive behaviour to short-term fluctuations in

renewable energy availability, the green-k8s-scheduler which picks up the descheduled Pods has to be configured accordingly, thus it runs in mode "S". The interval in which the green-k8s-descheduler checks the cluster state and triggers a Pod eviction process if needed is set to ten minutes. The green-k8s-descheduler is evaluated against the pure use of the green-k8s-scheduler, since the default scheduler has already been outperformed in the past tests.

5.3.1 Single Pod Descheduling

15.07.2020 06:00 a.m. - 14 Hours - Single Pod - Mode S

In the first test, one Pod will be deployed to the cluster. Care was taken to evaluate a period with a variety of renewable energy availability over time, where different nodes provide the best coverage for certain time frames respectively.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.14: Energy Consumption and Production with instant Workload - green-k8s-scheduler

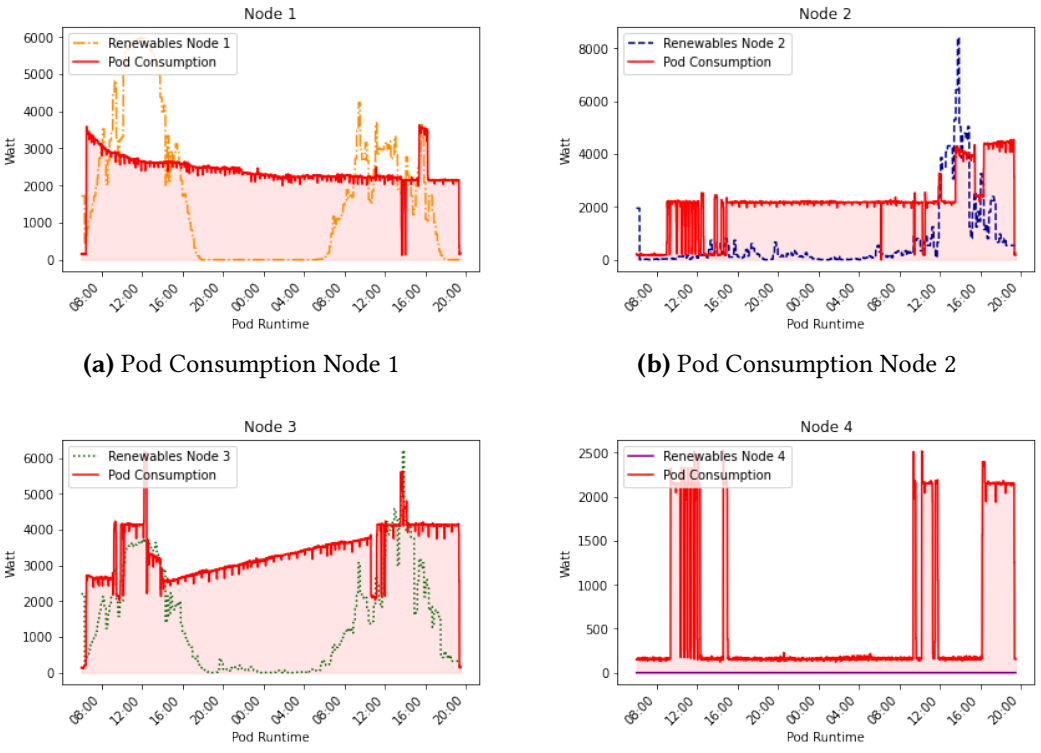
Figure 5.14 shows that after the initial scheduling to Node 3, the green-k8s-descheduler is triggered as soon as a higher share is available on Node 1, which leads to an eviction of the Pod and a rescheduling by the green-k8s-scheduler. The Pod stays at Node 1 until a node with a lower deficit or better coverage can be found. A fair amount of six green-k8s-descheduler

interventions over 14 hours runtime led to an renewable energy utilization of 81.3% of the given workload consumption. In contrast, at the node that would have been selected solely by the green-k8s-scheduler in mode "XL", a renewable energy coverage of 71.9% would have been achieved. Thus, the use of the green-k8s-descheduler led to an additional increase of 13.1%, which indicates an even higher optimization potential than evaluated in Section 5.2.

5.3.2 Multi Pod Descheduling

22.09.2020 06:00 a.m. - 38 Hours - Multiple Pods - Mode S

In the last test, the combination of the green-k8s-descheduler and the green-k8s-scheduler handling multiple Pods should be examined. Therefore, a total of eight Pods will be deployed instantly to the cluster at the beginning of the test period.



(a) Pod Consumption Node 1

(b) Pod Consumption Node 2

(c) Pod Consumption Node 3

(d) Pod Consumption Node 4

Fig. 5.15: Energy Consumption and Production with instant Workload - green-k8s-scheduler

Since we learned from Section 5.2.2 a) that due to balanced resource priorities the green-k8s-scheduler decisions are overruled at some point, the weight of the extender score among the default priority functions is increased from one to three during the test of the descheduler.

This is intended to favor the renewable energy availability over balanced node resource consumption and thus avoid descheduling-scheduling loops.

Figure 5.15 shows that after the initial scheduling of five Pods to Node 1 and three Pods to Node 3, which provide the highest renewable share at the time, the green-k8s-descheduler is triggered at around 10:00 a.m. to deschedule workload from Node 3 with the intention to reschedule it at Node 1. However, the increased weight of the green-k8s-scheduler extender does not seem to overrule the balanced resource usage and service spread priorities, which can be obtain from multiple descheduling attempts at Node 2 and Node 4, where Pods are scheduled instead, despite no availability of renewable energy resources. This result suggests that the weight has to be configured accordingly to avoid false placements and the resulting scheduling-descheduling loop. Nevertheless, for the majority of the runtime and with regards to the overall Pod distribution, the workload seems to match the renewable energy availability properly, e.g. by following the two peaks at Node 3 and the higher utilization of Node 2 towards the end of the runtime. The use of the descheduler results in a coverage of the given workload by renewable energy of 36.1%. Compared to the green-k8s-scheduler in "XL" mode with standard priorities and weighting, which achieves a 33.0% coverage, this represents an increase of 9.4%. In this case, both variants deliver a satisfactory result, while the descheduler can still achieve a slightly higher coverage thanks to the adjusted configuration. Thus, for workloads that are not adversely affected by Pod interruptions, the use if the green-k8s-descheduler is a good way to achieve a slightly higher utilization of renewable energy than with the green-k8s-scheduler. Additional fine tuning of the scheduler configuration is expected to unleash further potentials.

6 Conclusion

In this thesis, an annotation-based approach to enhance geo-distributed Kubernetes clusters with renewable-aware scheduling features was introduced. Therefore, renewable energy appliances were simulated by the green-k8s-energy-management and respective energy data was written into node annotations to be used by the green-k8s-scheduler and green-k8s-descheduler. This approach has not been used in any known work to date. As an extension to the Kubernetes scheduler, the green-k8s-scheduler offers a scheduling algorithm, which takes not only the nodes current renewable energy availability into account, but also optimizes Pod placement decisions with regard to the expected application runtime by involving renewable energy forecasts as well. This also distinguishes the green-k8s-scheduler from the default kube-scheduler as well as other known Kubernetes schedulers [20, 21], which solely focus on present availability at most. Five configuration options enable the green-k8s-scheduler to be tailored to different degrees of short, medium, long or continuously running applications, while the renewable energy utilization can be further optimized with the use of the green-k8s-descheduler. As a Pod running in the kube-system namespace, the green-k8s-descheduler is able to evict Pods from the cluster which can be rescheduled to more sustainable nodes if weather conditions change. In various test cases, the components were evaluated with a different number of Pods, different Pod runtimes and varying weather conditions / availability of renewable energy. Compared to the default scheduler, an increase in renewable energy utilization of up to 116% has been achieved. The use of Kubernetes as a basis also turned out to be advantageous, as the technology has already been widely adopted in modern cloud computing and thus the green-k8s-scheduler shows great potential for an easy deployment in productive systems.

In future research, several topics for further improvement of the green-k8s scheduler and the integration of renewable energy in distributed systems could be of interest. To gain efficiency and perform even better data-driven scheduling decisions, additional metrics could be included in the scheduler and deschedulers algorithms as well. Therefore, a variety of options are available, e.g. the charge level of batteries that can be integrated into the local energy management systems, environmental measurements like the temperature, to consider costs of

cooling or in the case of the green-k8s-descheduler, the cost of workload migrations from an energy consumption and quality of service perspective. In general, the integration of further data from heterogeneous sources is very easy with the annotation-based approach as used in this thesis. Furthermore, as addressed in existing research [115, 122], a complementation of spatial with temporal workload shifting for time-insensitive computing tasks could lead to a greater flexibility in terms of energy consumption in the cluster, thus an even more fine grained match of supply and demand of renewable energy could be achieved.

Bibliography

1. IPCC: *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]* Intergovernmental Panel on Climate Change, 2014. Available also from: https://www.ipcc.ch/site/assets/uploads/2018/02/SYR_AR5_FINAL_full.pdf. Accessed: 15.07.2021.
2. AMPONSAH, Nana Yaw et al.: Greenhouse gas emissions from renewable energy sources: A review of lifecycle considerations. *Renewable and Sustainable Energy Reviews*. 2014, vol. 39, pp. 461–475. ISSN 1364-0321. Available from DOI: https://doi.org/10.1016/j.rser.2014.07.087.
3. MASANET, Eric et al.: Recalibrating global data center energy-use estimates. *Science*. 2020, vol. 367, no. 6481, pp. 984–986. Available from DOI: 10.1126/science.aba3758.
4. IEA: *Data Centres and Data Transmission Networks* [https://www.iea.org/reports/data-centres-and-data-transmission-networks]. IEA, 2021. Accessed: 15.11.2021.
5. CNCF: *CNCF Survey 2019* [https://www.cncf.io/wp-content/uploads/2020/08/CNCF_Survey_Report.pdf]. Cloud Native Computing Foundation, 2019. Accessed: 15.11.2021.
6. GRANGE, Léo et al.: Green IT scheduling for data center powered with renewable energy. *Future Generation Computer Systems*. 2018, vol. 86, pp. 99–120. ISSN 0167-739X. Available from DOI: https://doi.org/10.1016/j.future.2018.03.049.
7. DENG, Xiang et al.: Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters. *IEEE Systems Journal*. 2016, vol. 10, no. 1, pp. 78–87. Available from DOI: 10.1109/JSYST.2014.2344028.
8. QU, Xiaoyang et al.: GreenMatch: Renewable-Aware Workload Scheduling for Massive Storage Systems. In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2016, pp. 403–412. Available from DOI: 10.1109/IPDPS.2016.24.
9. CHEN, Changbing et al.: Green-aware workload scheduling in geographically distributed data centers. In: *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*. 2012, pp. 82–89. Available from DOI: 10.1109/CloudCom.2012.6427545.

10. KUMAR, Neeraj et al.: Renewable Energy-Based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers. *IEEE Transactions on Industrial Informatics*. 2019, vol. 15, no. 5, pp. 2947–2957. Available from DOI: 10.1109/TII.2018.2800693.

11. GOIRI, Íñigo et al.: GreenSlot: Scheduling energy consumption in green datacenters. In: *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 2011, pp. 1–11. Available from DOI: 10.1145/2063384.2063411.

12. XU, Chenhan et al.: Renewable Energy-Aware Big Data Analytics in Geo-Distributed Data Centers with Reinforcement Learning. *IEEE Transactions on Network Science and Engineering*. 2020, vol. 7, no. 1, pp. 205–215. Available from DOI: 10.1109/TNSE.2018.2813333.

13. ZHOU, Zhi et al.: Carbon-Aware Load Balancing for Geo-distributed Cloud Services. In: *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. 2013, pp. 232–241. Available from DOI: 10.1109/MASCOTS.2013.31.

14. HAN, Qiaoni et al.: Energy-Aware and QoS-Aware Load Balancing for HetNets Powered by Renewable Energy. *Computer Networks*. 2015, vol. 94. Available from DOI: 10.1016/j.comnet.2015.10.025.

15. WANG, Xiaoying et al.: A green-aware virtual machine migration strategy for sustainable datacenter powered by renewable energy. *Simulation Modelling Practice and Theory*. 2015, vol. 58, pp. 3–14. ISSN 1569-190X. Available from DOI: https://doi.org/10.1016/j.simpat.2015.01.005. Special Issue on TECHNIQUES AND APPLICATIONS FOR SUSTAINABLE ULTRASCALE COMPUTING SYSTEMS.

16. LI, Chao et al.: Managing Green Datacenters Powered by Hybrid Renewable Energy Systems. In: *11th International Conference on Autonomic Computing (ICAC 14)*. Philadelphia, PA: USENIX Association, 2014, pp. 261–272. ISBN 978-1-931971-11-9. Available also from: http://www.usenix.org/conference/icac14/technical-sessions/presentation/li_chao.

17. FAN, Qiang et al.: Energy Driven Avatar Migration in Green Cloudlet Networks. *IEEE Communications Letters*. 2017, vol. 21, no. 7, pp. 1601–1604. Available from DOI: 10.1109/LCOMM.2017.2684812.

18. GUO, Yuanxiong et al.: Energy and Network Aware Workload Management for Sustainable Data Centers with Thermal Storage. *IEEE Transactions on Parallel and Distributed Systems*. 2014, vol. 25, no. 8, pp. 2030–2042. Available from DOI: 10.1109/TPDS.2013.278.

19. KHOSRAVI, Atefeh et al.: Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers. *Concurrency and Computation: Practice and Experience*. 2017, vol. 29, no. 18, e4125. Available from DOI: https://doi.org/10.1002/cpe.4125. e4125 cpe.4125.

20. JAMES, Aled; SCHIEN, Daniel: A Low Carbon Kubernetes Scheduler. In: *ICT4S*. 2019.

21. KAUR, Kuljeet et al.: KEIDS: Kubernetes-Based Energy and Interference Driven Scheduler for Industrial IoT in Edge-Cloud Ecosystem. *IEEE Internet of Things Journal*. 2020, vol. 7, no. 5, pp. 4228–4237. Available from DOI: 10.1109/JIOT.2019.2939534.

22. GOODMAN, Amy: *School Strike for Climate: Meet 15-Year-Old Activist Greta Thunberg, Who Inspired a Global Movement* [https://www.democracynow.org/2018/12/11/meet_the_15_year_old_swedish]. democracynow.org, 2018. Accessed: 15.07.2021.

23. BORDUAS, Nadine; DONAHUE, Neil M.: Chapter 3.1 - The Natural Atmosphere. In: TÖRÖK, Béla; DRANSFIELD, Timothy (eds.). *Green Chemistry*. Elsevier, 2018, pp. 131–150. ISBN 978-0-12-809270-5. Available from DOI: https://doi.org/10.1016/B978-0-12-809270-5.00006-6.

24. SCHEWE, Jacob; EDENHOFER, Ottmar: *Climate change: Physics and impacts* [Economics of Climate Change Lecture, TU Berlin]. 2019. Attended: Summer Term 2019.

25. STERN, Nicholas: The Economics of Climate Change. *The American Economic Review*. 2008, vol. 98, no. 2, pp. 1–37. ISSN 00028282. Available also from: http://www.jstor.org/stable/29729990.

26. BUIS, Alan: *The Atmosphere: Getting a Handle on Carbon Dioxide* [https://climate.nasa.gov/news/2915/the-atmosphere-getting-a-handle-on-carbon-dioxide/]. NASA's Jet Propulsion Laboratory, 2019. Accessed: 15.07.2021.

27. ETMINAN, M. et al.: Radiative forcing of carbon dioxide, methane, and nitrous oxide: A significant revision of the methane radiative forcing. *Geophysical Research Letters*. 2016, vol. 43, no. 24, pp. 12, 614–12, 623. Available from DOI: https://doi.org/10.1002/2016GL071930.

28. ROCKSTRÖM, Johan et al.: A safe operating space for humanity. *Nature*. 2009, vol. 461, no. 7263, pp. 472–475. ISSN 1476-4687. Available from DOI: 10.1038/461472a.

29. STEFFEN, Will et al.: Planetary boundaries: Guiding human development on a changing planet. *Science*. 2015, vol. 347, no. 6223. ISSN 0036-8075. Available from DOI: 10.1126/science.1259855.

30. IPCC: *About the IPCC* [https://www.ipcc.ch/about/]. Intergovernmental Panel on Climate Change, [n.d.]. Accessed: 15.07.2021.

31. NOAA: *Trends in Atmospheric Carbon Dioxide* [https://www.esrl.noaa.gov/gmd/ccgg/trends/weekly.html]. National Oceanic and Atmospheric Administration, 2020. Accessed: 15.07.2021.

32. LINDSEY, Rebecca: *Climate Change: Atmospheric Carbon Dioxide* [https://www.climate.gov/news-features/understanding-climate/climate-change-atmospheric-carbon-dioxide]. National Oceanic and Atmospheric Administration, 2020. Accessed: 15.07.2021.

33. NOAA: *Annual Mean Growth Rate for Mauna Loa, Hawaii* [https://www.esrl.noaa.gov/gmd/ccgg/trends/gr.html]. National Oceanic and Atmospheric Administration, 2020. Accessed: 15.07.2021.

34. CANADELL, Josep G. et al.: Contributions to accelerating atmospheric CO2 growth from economic activity, carbon intensity, and efficiency of natural sinks. *Proceedings of the National Academy of Sciences*. 2007, vol. 104, no. 47, pp. 18866–18870. ISSN 0027-8424. Available from DOI: 10.1073/pnas.0702737104.

35. LENTON, Timothy M. et al.: Tipping elements in the Earth's climate system. *Proceedings of the National Academy of Sciences*. 2008, vol. 105, no. 6, pp. 1786–1793. ISSN 0027-8424. Available from DOI: 10.1073/pnas.0705414105.

36. PARMESAN, Camille: Ecological and Evolutionary Responses to Recent Climate Change. *Annual Review of Ecology, Evolution, and Systematics*. 2006, vol. 37, no. 1, pp. 637–669. Available from DOI: 10.1146/annurev.ecolsys.37.091305.110100.

37. HOEGH-GULDBERG, Ove et al.: Coral Reef Ecosystems under Climate Change and Ocean Acidification. *Frontiers in Marine Science*. 2017, vol. 4, p. 158. ISSN 2296-7745. Available from DOI: 10.3389/fmars.2017.00158.

38. POWERS, Ryan P.; JETZ, Walter: Global habitat loss and extinction risk of terrestrial vertebrates under future land-use-change scenarios. *Nature Climate Change*. 2019, vol. 9, no. 4, pp. 323–329. ISSN 1758-6798. Available from DOI: 10.1038/s41558-019-0406-z.

39. STOTT, Peter: How climate change affects extreme weather events. *Science*. 2016, vol. 352, no. 6293, pp. 1517–1518. ISSN 0036-8075. Available from DOI: 10.1126/science.aaf7271.

40. VERMEER, Martin; RAHMSTORF, Stefan: Global sea level linked to global temperature. *Proceedings of the National Academy of Sciences*. 2009, vol. 106, no. 51, pp. 21527–21532. ISSN 0027-8424. Available from DOI: 10.1073/pnas.0907765106.

41. NEUMANN, Barbara et al.: Future Coastal Population Growth and Exposure to Sea-Level Rise and Coastal Flooding - A Global Assessment. *PLOS ONE*. 2015, vol. 10, no. 3, pp. 1–34. Available from DOI: 10.1371/journal.pone.0118571.

42. HASEGAWA, Tomoko et al.: Risk of increased food insecurity under stringent global climate change mitigation policy. *Nature Climate Change*. 2018, vol. 8, no. 8, pp. 699–703. ISSN 1758-6798. Available from DOI: 10.1038/s41558-018-0230-x.

43. SCHEWE, Jacob et al.: Multimodel assessment of water scarcity under climate change. *Proceedings of the National Academy of Sciences*. 2014, vol. 111, no. 9, pp. 3245–3250. ISSN 0027-8424. Available from DOI: 10.1073/pnas.1222460110.

44. FAO: *The impact of disasters and crises on agriculture and food security*. Food and Agriculture Organization of the United Nations, 2021. Available from DOI: 10.4060/cb3673en.

45. MISRA, Anil Kumar: Climate change and challenges of water and food security. *International Journal of Sustainable Built Environment*. 2014, vol. 3, no. 1, pp. 153–165. ISSN 2212-6090. Available from DOI: https://doi.org/10.1016/j.ijsbe.2014.04.006.

46. MACH, Katharine J. et al.: Climate as a risk factor for armed conflict. *Nature*. 2019, vol. 571, no. 7764, pp. 193–197. ISSN 1476-4687. Available from DOI: 10.1038/s41586-019-1300-6.

47. BROWN, Oli: *Migration and Climate Change*. United Nations, 2008. Available from DOI: 10.18356/26de4416-en.

48. UNHCR: *Displaced on the frontlines of the climate emergency* [https://storymaps.arcgis.com/stories/065d18218b654c798ae9f360a626d903]. UNHCR, The UN Refugee Agency, 2021. Accessed: 15.07.2021.

49. HALLEGATTE, Stephane et al.: Future flood losses in major coastal cities. *Nature Climate Change*. 2013, vol. 3, no. 9, pp. 802–806. ISSN 1758-6798. Available from DOI: 10.1038/nclimate1979.

50. CISSÉ, Guéladio: Food-borne and water-borne diseases under climate change in low- and middle-income countries: Further efforts needed for reducing environmental health exposure risks. *Acta Tropica*. 2019, vol. 194, pp. 181–188. ISSN 0001-706X. Available from DOI: https://doi.org/10.1016/j.actatropica.2019.03.012.

51. ROCKLÖV, Joacim; DUBROW, Robert: Climate change: an enduring challenge for vector-borne disease prevention and control. *Nature Immunology*. 2020, vol. 21, no. 5, pp. 479–483. ISSN 1529-2916. Available from DOI: 10.1038/s41590-020-0648-y.

52. LUBER, George; MCGEEHIN, Michael: Climate Change and Extreme Heat Events. *American Journal of Preventive Medicine*. 2008, vol. 35, no. 5, pp. 429–435. ISSN 0749-3797. Available from DOI: https://doi.org/10.1016/j.amepre.2008.08.021. Theme Issue: Climate Change and the Health of the Public.

53. KEMFERT, Claudia: The Economic Costs of Climate Change. *Weekly Report*. 2005, vol. 1, no. 2, pp. 43–49. ISSN 1860-3343. Available also from: http://hdl.handle.net/10419/150978.

54. BURKE, Marshall et al.: Large potential reduction in economic damages under UN mitigation targets. *Nature*. 2018, vol. 557, no. 7706, pp. 549–553. ISSN 1476-4687. Available from DOI: 10.1038/s41586-018-0071-9.

55. KJELLSTROM, Tord et al.: Heat, Human Performance, and Occupational Health: A Key Issue for the Assessment of Global Climate Change Impacts. *Annual Review of Public Health*. 2016, vol. 37, no. 1, pp. 97–112. Available from DOI: 10.1146/annurev-publhealth-032315-021740. PMID: 26989826.

56. DIFFENBAUGH, Noah S.; BURKE, Marshall: Global warming has increased global economic inequality. *Proceedings of the National Academy of Sciences*. 2019, vol. 116, no. 20, pp. 9808–9813. ISSN 0027-8424. Available from DOI: 10.1073/pnas.1816020116.

57. SCHEFFERS, Brett R. et al.: The broad footprint of climate change from genes to biomes to people. *Science*. 2016, vol. 354, no. 6313. ISSN 0036-8075. Available from DOI: 10.1126/science.aaf7671.

58. WHO: *Climate change and health* [https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health]. World Health Organization, 2018. Accessed: 15.07.2021.

59. BMU: *United Nations Framework Convention on Climate Change (UNFCCC)* [https://www.bmu.de/en/topics/climate-energy/climate/international-climate-policy/united-nations-framework-convention-on-climate-change-unfccc/]. Federal Ministry for the Environment, Nature Conservation and Nuclear Safety, [n.d.]. Accessed: 15.07.2021.

60. UNFCCC: *What is the United Nations Framework Convention on Climate Change?* [https://unfccc.int/process-and-meetings/the-convention/what-is-the-united-nations-framework-convention-on-climate-change]. [N.d.]. Accessed: 15.07.2021.

61. UNITED NATIONS: *The Paris Agreement* [https://sustainabledevelopment.un.org/topics/climatechange]. [N.d.]. Accessed: 15.07.2021.

62. UNFCCC: *The Paris Agreement* [https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement]. [N.d.]. Accessed: 15.07.2021.

63. UNFCCC: *The Paris Agreement* [https://unfccc.int/resource/docs/2015/cop21/eng/10.pdf]. 2016. Accessed: 15.07.2021.

64. BARRETT, Scott; DANNENBERG, Astrid: Climate negotiations under scientific uncertainty. *Proceedings of the National Academy of Sciences of the United States of America*. 2012/10/08. 2012, vol. 109, no. 43, pp. 17372–17376. ISSN 1091-6490. Available from DOI: 10.1073/pnas.1208417109. 1208417109[PII].

65. DUSO, Marco et al.: *Why Climate Change Is No Prisoner's Dilemma* [https://www.bcg.com/de-de/publications/2020/short-term-solution-to-tackle-climate-change]. Boston Consulting Group, 2020. Accessed: 15.07.2021.

66. BLÜMM, FLORIAN: *CO2 pro kWh: Welche ist die klimafreundlichste Energiequelle?)* [https://www.tech-for-future.de/co2-kwh-strom/]. 2021. Accessed: 15.10.2021.

67. BLÜMM, FLORIAN: *Vollkosten pro kWh: Welche ist die günstigste Energiequelle?)* [https://www.tech-for-future.de/kosten-kwh]. 2021. Accessed: 15.10.2021.

68. STATISTISCHES BUNDESAMT: *Stromerzeugung 2020* [https://www.destatis.de/DE/Presse/Pressemitteilungen/2021/03/PD21_101_43312.html]. Statistisches Bundesamt, 2021. Accessed: 15.11.2021.

69. BMU: *Climate Action Plan 2050 – Germany's long-term low greenhouse gas emission development strategy* [https://www.bmu.de/en/topics/climate-energy/climate/national-climate-policy/greenhouse-gas-neutral-germany-2050/]. [N.d.]. Accessed: 15.07.2021.

70. THE FEDERAL GOVERNMENT OF GERMANY: *Climate Action Programme 2030* [https://www.bundesregierung.de/breg-en/issues/climate-action]. [N.d.]. Accessed: 15.07.2021.

71. EUROPEAN COMMISSION: *EU climate action and the European Green Deal* [https://ec.europa.eu/clima/policies/eu-climate-action_en]. European Commission, [n.d.]. Accessed: 15.07.2021.

72. EUROPEAN COMMISSION: *EU Emissions Trading System (EU ETS)* [https://ec.europa.eu/clima/policies/ets_en]. European Commission, [n.d.]. Accessed: 15.07.2021.

73. EUROPEAN COMMISSION: *Emissions cap and allowances)* [https://ec.europa.eu/clima/policies/ets/cap_en]. European Commission, [n.d.]. Accessed: 15.07.2021.

74. EUROPEAN COMMISSION: *EDGAR - Emissions Database for Global Atmospheric Research* [https://edgar.jrc.ec.europa.eu/]. European Commission, [n.d.]. Accessed: 15.07.2021.

75. ICAP: *Emissions Trading Worldwide: Status Report 2020* [https://icapcarbonaction.com/en/?option=com_attach&task=download&id=677]. Berlin: International Carbon Action Partnership, 2020. Accessed: 15.07.2021.

76. LARANJA RIBEIRO, Marcos Paulo et al.: Adoption phases of Green Information Technology in enhanced sustainability: A bibliometric study. *Cleaner Engineering and Technology*. 2021, vol. 3, p. 100095. ISSN 2666-7908. Available from DOI: https://doi.org/10.1016/j.clet.2021.100095.

77. EL-HAWARY, Mohamed E.: The Smart Grid—State-of-the-art and Future Trends. *Electric Power Components and Systems*. 2014, vol. 42, no. 3-4, pp. 239–250. Available from DOI: 10.1080/15325008.2013.868558.

78. HERWEIJER, Celine; AZHAR, Azeem: *The State of Climate Tech 2020*. PriceWaterhouseCooper, 2020. Available also from: https://www.pwc.com/gx/en/services/sustainability/assets/pwc-the-state-of-climate-tech-2020.pdf.

79. TRUEMAN, Charlotte: *What impact are data centres having on climate change?* 2021. Available also from: https://www.computerworld.com/article/3431148/why-data-centres-are-the-new-frontier-in-the-fight-against-climate-change.html.

80. SHUJA, Junaid et al.: Sustainable Cloud Data Centers: A survey of enabling techniques and technologies. *Renewable and Sustainable Energy Reviews*. 2016, vol. 62, pp. 195–214. ISSN 1364-0321. Available from DOI: https://doi.org/10.1016/j.rser.2016.04.034.

81. JONES, Nicola: How to stop data centres from gobbling up the world's electricity. *Nature*. 2018, vol. 561, pp. 163–166. Available from DOI: 10.1038/d41586-018-06610-y.

82. ANDRAE, Anders: New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letters*. 2020, vol. 3, pp. 19–31. Available from DOI: 10.30538/psrp-easl2020.0038.

83. MELL, Peter; GRANCE, Timothy: *The NIST Definition of Cloud Computing*. Gaithersburg, MD, 2011-09. Tech. rep., 800-145. National Institute of Standards and Technology (NIST). Available also from: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

84. GIBSON, Joel et al.: Benefits and challenges of three cloud computing service models. In: *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*. 2012, pp. 198–205. Available from DOI: 10.1109/CASoN.2012.6412402.

85. DIGNAN, LARRY: *Top cloud providers: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players* [https://www.zdnet.com/article/the-top-cloud-providers-of-2021-aws-microsoft-azure-google-cloud-hybrid-saas/]. ZDNet.com, 2021. Accessed: 02.01.2022.

86. CAO, Hung et al.: Analytics Everywhere: Generating Insights From the Internet of Things. *IEEE Access*. 2019, vol. 7, pp. 71749–71769. Available from DOI: 10.1109/ACCESS.2019.2919514.

87. QAYYUM, Tariq et al.: FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment. *IEEE Access*. 2018, vol. 6, pp. 63570–63583. Available from DOI: 10.1109/ACCESS.2018.2877696.

88. YOUSEFPOUR, Ashkan et al.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*. 2019, vol. 98, pp. 289–330. ISSN 1383-7621. Available from DOI: https://doi.org/10.1016/j.sysarc.2019.02.009.

89. CISCO: *Cisco Annual Internet Report* [https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf]. Cisco, 2021. Accessed: 02.01.2022.

90. IBM CLOUD EDUCATION: *Virtualization* [https://www.ibm.com/cloud/learn/virtualization-a-complete-guide]. IBM, 2019. Accessed: 02.01.2022.

91. KUBERNETES: *What is Kubernetes?)* [https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/]. [N.d.]. Accessed: 15.07.2021.

92. REDHAT: *Containers vs VMs* [https://www.redhat.com/en/topics/containers/containers-vs-vms]. RedHat, 2020. Accessed: 02.01.2022.

93. CLOUD NATIVE COMPUTING FOUNDATION: *CNCF SURVEY 2020* [https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf]. Cloud Native Computing Foundation, 2020. Accessed: 02.01.2022.

94. REDHAT: *Kubernetes adoption, security, and market trends report 2021* [https://www.redhat.com/en/resources/kubernetes-adoption-security-market-trends-2021-overview]. RedHat, 2021. Accessed: 02.01.2022.

95. DATADOG: *11 Facts about real world container use* [https://www.datadoghq.com/container-report-2020/]. Datadog, 2020. Accessed: 02.01.2022.

96. BURNS, Brendan et al.: Borg, Omega, and Kubernetes: Lessons Learned from Three Container-Management Systems over a Decade. *Queue*. 2016, vol. 14, no. 1, pp. 70–93. ISSN 1542-7730. Available from DOI: 10.1145/2898442.2898444.

97. BITTENCOURT, Luiz F. et al.: Scheduling in distributed systems: A cloud computing perspective. *Computer Science Review*. 2018, vol. 30, pp. 31–54. ISSN 1574-0137. Available from DOI: https://doi.org/10.1016/j.cosrev.2018.08.002.

98. WOJCIECHOWSKI, Łukasz et al.: NetMARKS: Network Metrics-AwaRe Kubernetes Scheduler Powered by Service Mesh. In: *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. 2021, pp. 1–9. Available from DOI: 10.1109/INFOCOM42981.2021.9488670.

99. HUAXIN, Shi et al.: An Improved Kubernetes Scheduling Algorithm for Deep Learning Platform. In: *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 2020, pp. 113–116. Available from DOI: 10.1109/ICCWAMTIP51612.2020.9317317.

100. BELTRE, Angel et al.: KubeSphere: An Approach to Multi-Tenant Fair Scheduling for Kubernetes Clusters. In: *2019 IEEE Cloud Summit*. 2019, pp. 14–20. Available from DOI: 10.1109/CloudSummit47114.2019.00009.

101. SANTOS, José et al.: Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications. In: *2019 IEEE Conference on Network Softwarization (NetSoft)*. 2019, pp. 351–359. Available from DOI: 10.1109/NETSOFT.2019.8806671.

102. ROSSI, Fabiana et al.: Geo-distributed efficient deployment of containers with Kubernetes. *Computer Communications*. 2020, vol. 159, pp. 161–174. ISSN 0140-3664. Available from DOI: https://doi.org/10.1016/j.comcom.2020.04.061.

103. SONG, Shengbo et al.: Gaia Scheduler: A Kubernetes-Based Scheduler Framework. In: *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. 2018, pp. 252–259. Available from DOI: 10.1109/BDCloud.2018.00048.

104. YANG, Ying; CHEN, Lijun: Design of Kubernetes Scheduling Strategy Based on LSTM and Grey Model. In: *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. 2019, pp. 701–707. Available from DOI: 10.1109/ISKE47853.2019.9170419.

105. FU, Yuqi et al.: Progress-based Container Scheduling for Short-lived Applications in a Kubernetes Cluster. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 278–287. Available from DOI: 10.1109/BigData47090.2019.9006427.

106. TOWNEND, Paul et al.: Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes. In: *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 2019, pp. 156–15610. Available from DOI: 10.1109/SOSE.2019.00030.

107. KATENBRINK, Florian et al.: Dynamic Scheduling for Seamless Computing. In: *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*. 2018, pp. 41–48. Available from DOI: 10.1109/SC2.2018.00013.

108. CHIMA OGBUACHI, Michael et al.: Context-Aware K8S Scheduler for Real Time Distributed 5G Edge Computing Applications. In: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2019, pp. 1–6. Available from DOI: 10.23919/SOFTCOM.2019.8903766.

109. RAUSCH, Thomas et al.: Optimized container scheduling for data-intensive serverless edge computing. *Future Generation Computer Systems*. 2021, vol. 114, pp. 259–271. ISSN 0167-739X. Available from DOI: https://doi.org/10.1016/j.future.2020.07.017.

110. XING, Sikai et al.: A QoS-oriented Scheduling and Autoscaling Framework for Deep Learning. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–8. Available from DOI: 10.1109/IJCNN.2019.8852319.

111. MANIC, Milos et al.: Building Energy Management Systems: The Age of Intelligent and Adaptive Buildings. *IEEE Industrial Electronics Magazine*. 2016, vol. 10, no. 1, pp. 25–39. Available from DOI: 10.1109/MIE.2015.2513749.

112. KORONEN, Carolina et al.: Data centres in future European energy systems—energy efficiency, integration and policy. *Energy Efficiency*. 2020, vol. 13, no. 1, pp. 129–144. ISSN 1570-6478. Available from DOI: 10.1007/s12053-019-09833-8.

113. INMAN, Rich H. et al.: Solar forecasting methods for renewable energy integration. *Progress in Energy and Combustion Science*. 2013, vol. 39, no. 6, pp. 535–576. ISSN 0360-1285. Available from DOI: https://doi.org/10.1016/j.pecs.2013.06.002.

114. TIAN, Tian; CHERNYAKHOVSKIY, Ilya: *Forecasting Wind and Solar Generation: Improving System Operations* [https://www.nrel.gov/docs/fy16osti/65728.pdf]. National Renewable Energy Laboratory, 2016. Accessed: 23.11.2021.

115. WIESNER, Philipp et al.: Let's Wait Awhile: How Temporal Workload Shifting Can Reduce Carbon Emissions in the Cloud. In: *Proceedings of the 22nd International Middleware Conference*. Québec city, Canada: Association for Computing Machinery, 2021, pp. 260–272. Middleware '21. ISBN 9781450385343. Available from DOI: 10.1145/3464298.3493399.

116. BARROSO, Luiz André; HÖLZLE, Urs: The Case for Energy-Proportional Computing. *IEEE Computer*. 2007, vol. 40. Available also from: http://www.computer.org/portal/site/computer/index.jsp?pageID=computer_level1&path=computer/homepage/Dec07&file=feature.xml&xsl=article.xsl.

117. DARGIE, Waltenegus: A Stochastic Model for Estimating the Power Consumption of a Processor. *IEEE Transactions on Computers*. 2014, vol. 64. Available from DOI: 10.1109/TC.2014.2315629.

118. NGUYEN, Duong Tung; LE, Long Bao: Optimal Bidding Strategy for Microgrids Considering Renewable Energy and Building Thermal Dynamics. *IEEE Transactions on Smart Grid*. 2014, vol. 5, no. 4, pp. 1608–1620. Available from DOI: 10.1109/TSG.2014.2313612.

119. EUPD RESEARCH: *89 Prozent des Solarpotenzials auf deutschen Ein- und Zweifamilienhäusern sind noch ungenutzt* [https://www.eupd-research.com/89-prozent-des-solarpotenzials-noch-ungenutzt/]. EUPD Research, 2021. Accessed: 01.11.2021.

120. GREEN, Martin et al.: Solar cell efficiency tables (version 57). *Progress in Photovoltaics: Research and Applications*. 2021, vol. 29, no. 1, pp. 3–15. Available from DOI: https://doi.org/10.1002/pip.3371.

121. ANI, Samuel O. et al.: Comparison of Energy Yield of Small Wind Turbines in Low Wind Speed Areas. *IEEE Transactions on Sustainable Energy*. 2013, vol. 4, no. 1, pp. 42–49. Available from DOI: 10.1109/TSTE.2012.2197426.

122. RADOVANOVIC, Ana et al.: *Carbon-Aware Computing for Datacenters*. 2021. Available from arXiv: 2106.11750 [cs.DC].