Energy Efficiency in Web Development: Investigation of the Power Consumption of a Web Application with Different Load Distribution

Florian Kleinert and Hans-Knud Arndt

Abstract Almost all products, systems and activities leave an ecological footprint. For this reason, sustainability has become a central requirement of today's world. To date, this applies to software products only to a lesser extent. Nevertheless, recent developments such as the introduction of the Blue Angel as a label for energy- and resource-efficient software are continuously drawing more attention to this issue. One focus of research on this topic is the energy efficiency of these applications. Since software is used in different areas in many different architectures, a variety of factors influencing the respective power consumption are conceivable. Client-server architectures offer a special application case. Not only can the implementation of the algorithms influence the power consumption, but also the location where the computations are executed. This paper therefore investigates the influence of different load distributions on the energy efficiency of these architectures. The results obtained suggest that sometimes serious differences arise due to different calculation locations. These differences are due to multiple factors. Knowledge about this not only enables a more accurate assessment of the energy efficiency of web applications, but can also serve as a principle of sustainable web development and help developers to reduce the ecological footprint of their applications.

Keywords Green software · Green IT · Green by IT

1 Introduction

As a result of global climate change, an increasing interest in sustainability and energy efficiency has been developing worldwide in all areas of the economy for some time now. This has also increasingly brought the development of sustainable ICT applications to the fore, as these have a non-negligible impact on the environment

F. Kleinert · H.-K. Arndt (✉)
Otto von Guericke University Magdeburg, Universitätsplatz 2, 39016 Magdeburg, Germany
e-mail: hans-knud.arndt@iti.cs.uni-magdeburg.de

F. Kleinert
e-mail: florian.kleinert@st.ovgu.de

V. Wohlgemuth et al. (eds.), *Advances and New Trends in Environmental Informatics*,
Progress in IS, https://doi.org/10.1007/978-3-031-18311-9_12

[1]. The greatest focus here is on reducing energy consumption in addition to saving costs [2]. It is therefore essential to also consider the influence of the software. Although electricity is initially required for hardware, this consumption is caused by the associated software [3]. Nevertheless, it has only been in the last few years that attempts have been made to direct the focus of Green IT to the influence of those non-physical components. Although ongoing research has made hardware more and more efficient and thus able to handle the same algorithms with less power consumption, the energy consumption of the entire ICT sector continues to rise year after year [4]. Within this sector, much of the existing consumption is caused by Internet-based services [5]. For example, research has shown that the volume of data transported has increased by a factor of 4.5 within five years [6]. This increase can be explained by the rebound effect. This effect states that increased resource efficiency leads to increased demand and ultimately to less savings than expected [7]. This is a major contributor to the steadily increasing demand for energy. This is confirmed by research by Philippot et al. They found out that not only large differences exist in the energy demand of different websites, but that newer sites in particular consume more [8]. Therefore, more focus should be put on the development of resource-efficient web applications. Especially since even small changes in the source code can have a big impact on the energy efficiency [9, 10]. All these problems and developments should give software developers reason to pay more attention to and understand the impact of their code on the energy consumption of a system, since they are the ones who have a significant influence on the power consumption of their application.

The aim of this paper is to increase the understanding of the power consumption of a web application by trying to answer the often-asked question about the optimal load distribution between frontend and backend. The paper approaches this question from the perspective of energy efficiency. Possible degradations of the user experience due to potentially longer load times are not included in the results. Additionally, other factors for sustainable software products, apart from the power consumption of these, are not investigated. For this purpose, a client-server structure is developed, which is able to execute realistic workloads for web applications both on the client side and on the server side. This program can then be compared in both scenarios using a System Under Test (SUT) for energy efficiency. Furthermore, the values obtained this way should also allow to draw conclusions about practical use cases and enable a recommendation to be made for the real development of web applications. This approach is intended to answer the following research questions:

1. How great is the influence of load distribution on the energy efficiency of a web application?
2. Which factors are responsible for the differences that arise?
3. How does an optimal load distribution for web applications result in practical application scenarios?

2 State of the Art

2.1 Power Consumption of a Web Application

In order to understand the power consumption of a client-server architecture, it is first necessary to consider the areas in which power consumption occurs. On the one hand, the client and server systems themselves require power, but data transmission via the respective network and the corresponding visualization should not be neglected either. Since the hardware of a server that is relevant for power consumption differs from that of a client not by the components used, but usually only by their percentage of the total power requirement, these are merely classified in the following and there is no separation between client and server. This chapter is limited by the non-observance of power consumption due to non-resource-conserving behavior in handling the devices and other externally occurring consumption, such as the cooling of a server. Likewise, reasons for the static power consumption of the components mentioned, which are due to their construction or architecture, are not explained further. The component manufacturers are responsible for improvements in these areas. Thus, only components that are influenced by a shift in the load distribution are considered. Consumptions by monitors or user interfaces, for example, are therefore excluded. These restrictions are made because this type of consumption cannot be influenced by the corresponding software and is therefore not relevant for this study.

The CPU, the main memory and the hard disk mainly cause the total power consumption of a computer system. In addition, static energy consumption occurs in both servers and client computers due to the design of the components. This includes, for example, energy for the chipset or motherboard or losses in the power supply [11]. For web applications, the influence of the network must also be considered. Although all these components consume energy, it is still necessary to know their corresponding influence in order to be able to put the importance of the components in relation to each other. In addition, this allows a better assessment of the extent to which the individual components can be influenced by software.

It has already been proven by [12] that dynamic power consumption of an application is mainly caused by the processor and various storage media. The majority of this is also attributable to the processor. In the case of servers, it was found that one third of the total energy consumption can be attributed to the CPU. In contrast, this proportion is much lower for computers or laptops [12]. However, the exact amount depends on the system used and the components it contains as well as the software executed. For these reasons, the greatest focus should always be on the influence of processors when considering the energy efficiency of software.

Network power consumption

Determining the power consumption of network transmission is a very complex task. While many attempts have been made to estimate how much energy is consumed per gigabyte sent, unfortunately they vary widely not only in their methodology but also in their results. The results obtained are significantly influenced by the size of

the existing network, the year of the study and the assumptions made. The values obtained in each case differ by up to 7 kWh/GB [13]. However, an alignment of these calculations by Aslan showed that similar results can be obtained even with different methodologies. This alignment resulted in a consumption of 0.06 kWh per gigabyte transmitted in 2015, which is supported by several other studies. However, it was also possible to derive the trend that the power consumption caused per gigabyte sent on the Internet is approximately halved every two years [13]. Thus, the current energy consumption caused by data traffic during the use of a web application can unfortunately only be estimated. In general, however, it can be said that the power consumption in the network depends on the size of the data sent and increases with an increasing number of these or with an increasing number of calls to the page. To reduce this demand, the focus should first be on reducing the amount of data sent. Thus, the introduction and use of efficient data formats is a fundamental requirement for sustainable web applications [3]. In addition to this, Dick et. al already presented 12 principles with which the size of sent files can be effectively reduced [14]. Finally, it should be noted that even with decreasing consumption per gigabyte, the total consumption increases linearly with the number of users and can therefore quickly be the cause of considerable power consumption, especially for frequently used services.

Other power-consuming components

Due to their similar importance for the topic, the remaining components are summarized in the following chapter. Components of the respective system such as the ventilation, the chipset, the mainboard or also the power supply are considered here. Components of this type can be found on the client as well as on the respective servers. Their significance for power consumption is that the demand induced by them can be assigned almost exclusively to the static portion of the total consumption. Moreover, these components collectively consume most of the energy during a system's idle state. Although these components differ in their respective shares, none of them can directly be influenced by the user or by the software. In this case, only the corresponding manufacturers of the parts have an influence on their energy efficiency. In addition, it is partly not possible to separate the consumption of the individual components from each other due to the design. Likewise, this covers all components whose energy demand is not affected by a shift of the work expenditures. Due to these facts, they are not treated in more detail in this paper.

2.2 Measurement of Consumption

For a measurement of the power consumption of software, v. Kistowski already described four characteristics as quality criteria. These are reproducibility, fairness, verifiability as well as usability. In addition to this, the relevance of the workload plays an important role. All these should be followed for a successful measurement

of the energy demand of software [15, 16]. For this reason, the investigations carried out follow those criteria.

Metrics

Metrics form the basis of any comparison. For this reason, it should always be made clear before the start of an investigation, which evaluation criteria are to be used, and whether they are suitable for the scenario in question. This is particularly relevant for measurements of the energy efficiency of software. Up to now, software has been developed with a focus on the best possible performance. However, it has been found that optimizing software for energy efficiency is not necessarily compatible with the goal of increased performance [17]. For this reason, Johann et al. define a widely used metric in software energy efficiency as the quotient of work done and energy consumed [18]. However, a weakness of this metric is the fact that the energy efficiency of a system or an implementation is not a constant value. Rather, it additionally depends on the workload of the system [19]. Thus, it can be assumed that a heavily loaded system is more energy efficient than a less heavily loaded system. This is particularly relevant for an investigation of load distribution, since servers are normally much more heavily loaded than the client. For this reason, it is essential to simulate a realistic load for systems when conducting experiments.

3 Related Work

Due to the increasing importance of the sustainability of applications, the research carried out has already been able to build up a broad knowledge base. Publications on the energy efficiency of applications form a broad field of research. In particular, the focus on measurement techniques should be mentioned here. The findings from these studies show that by using a System Under Test, almost all applications can be investigated with only minimal adjustments in the test setup [3]. Due to the many conceivable areas of application for software, however, limitations for comparisons of different applications had to be determined first. Due to the manifold application areas and functionalities, only products with very similar or the same functionality can be compared with each other. Nonetheless, studies conducted under this assumption were able to identify significant differences. Exemplary for this are the findings of Kern, Guldner and Naumann, who could prove this by measurements of different text processing programs. Similar results have also been shown in measurements of other use cases, such as web browsers or content management systems [3].

Since power consumption by software only came into focus in the course of these publications, it was also necessary to gain more attention for this problem. In a survey of programmers, for example, Pang was able to determine that only 18 percent of the respondents evaluate the software they have developed on the basis of its energy efficiency, and that this is not a quality criterion frequently demanded by users or customers [20]. One possible reason for this is also the low level of knowledge about the sustainability of software products. The reasons for this can be multifaceted. On

the one hand, the sustainability of applications continues to be underrepresented in curricula [21], and on the other hand, there are hardly any fixed specifications which aspects are to be considered in the evaluation of sustainable software. Based on this motivation, Gröger et al. developed evaluation principles "for resource-efficient software" [3] on behalf of the Federal Environment Agency. In the further course, these also served as the basis for the creation of the Blue Angel eco-label for software. In the near future, users will be able to take this label into account in their decision-making, and the development of software will also receive clear indications in all areas of sustainability. This presumed behavior of users has already been confirmed in a publication by Kern, in which it was shown that a majority of users would prefer a product certified as sustainable to its non-certified counterpart, as long as it does not have any restrictions on functionality [22].

In this context, however, it is necessary to note that a power-saving implementation is an essential component of a sustainable application, but not the only one. For this reason, several studies with different approaches have tried to determine quality aspects. A common assumption is that a software product should have as little impact on the environment as possible over its entire life cycle [3]. Within this life cycle, additional consideration is given to the hardware used and its consumption of natural resources.

In order to consider as many of these factors as possible and also to support product managers, the development of frameworks for sustainable software development was another essential part of published work. A frequently used approach is the Greensoft model, which supports the development of a software product throughout its entire life cycle and thus includes developers, administrators, and the users of the software [3]. Here, both direct and indirect influences during the creation, maintenance, and use of the application are considered in four phases.

Although almost all of the above findings can also be applied to client-server architectures, there is a lack of specific factors that influence sustainability, particularly in this area. This also applies to the energy efficiency. For example, power consumption has so far only been a focus of development in battery-powered systems such as smartphones or laptops. Nevertheless, the use and development of web applications also creates an ecological footprint that must be reduced. In this area, in particular, there is currently a lack of both a certification such as the Blue Angel and guidelines or tools for developers.

4 Details of the Experimental Setup

4.1 Workload

The selection of the workload, which is executed on the examined system, is of great importance for the evaluation of the later results. The calculations performed should also be used frequently in the real application of the respective software product.

In order to examine differences of energy requirements with different load distribution of web applications, it applies additionally that selected work expenditures can be executed typically both on the client and on the server. Similarly, operations should be selected on the basis of the hardware components that they occupy. To determine the greatest possible differences, the largest source of dynamic energy demand, the processor, should be the most heavily loaded. For a web application, it is also advisable to simulate a number of simultaneously operating clients that is as close to reality as possible. All these requirements pursue the goal of making obtained measured values relevant and validatable. An example for such tasks can be found in the processing of lists. These can typically be sorted, filtered or searched both on the client side and on the server side. Another advantage is that these tasks can be performed with any number of elements within the list and can be applied to a wide variety of data types. They also offer further flexibility with respect to the exact implementation method, which in turn can utilize different hardware capacities. Since the scope of this work only covers the influence of load balancing and it cannot be assumed that trends found in this process change by selecting different algorithms, the implementation only covers sorting methods. Quicksort was chosen as the exact method because it is one of the most widely used sorting algorithms and thus represents a practical example.

For the selection of the data within the lists, however, it must be considered that these must be reproducible. Especially for search and sorting algorithms, special attention should therefore be paid to the fact that obtained results are not falsified by random influences, such as an already randomly correctly sorted list. Since there are sometimes large differences in complexity between the best and the worst possible case, this could endanger both the correctness of the values obtained and their reproducibility.

In order to exclude these random factors as best as possible, measurements of both configurations are always performed with the same starting position. In addition, this procedure is carried out for both calculation methods with identical lists and repeated for 10 different lists with 10,000 random integers each. This procedure makes it possible to compare the mean values of the corresponding measurement series of different configurations. Due to the unique usage method, this comparison can always take place per sorted list. According to the definition of energy efficiency, each sorted list thus constitutes the useful work performed. To avoid differences in static consumption due to separate applications for the respective configurations, only one application was developed that offers the possibility to set the desired configuration of the load distribution via the user interface.

An initial configuration is that the client receives the unprocessed list and performs all calculations itself, i.e. in the frontend. This results in significantly less data traffic on the network. In addition, the workload is shifted so that the server is less busy, while the client has to bear more of the workload. For the sake of simplicity, it is also assumed that the power requirements for the calculations running in the frontend are identical for each user. Within this scenario, it is also important to note that only a very short time is required to perform the sorting, even if a list of realistic size is selected. Therefore, based on the half-second readout interval of the meter, it is

possible that the expected peaks in energy demand will not be detected by the meter. For this reason, it is necessary to perform these workloads repeatedly. However, it is necessary that these calculations are not carried out directly one after the other as quickly as possible, otherwise unrealistic utilization of the hardware will occur. Thus, the execution of the repeated computations is limited by waiting times in the JavaScript code. This implementation method results in a constant electrical power, which can be used to calculate the exact amount of energy consumed per list.

A second implementation involves the server performing all calculations itself. For the client, this means that no processing of the lists received is necessary on its side and thus no dynamic power consumption is incurred on its side. However, for each sorting or filtering of this data, a request to the corresponding server is necessary. This results in increased power consumption due to network transmission. In addition, server capacities are significantly more heavily utilized, depending on the number of users. The simulation of these clients is taken over directly by the frontend, which sends an individually definable number of simultaneous requests. This step examines how much the energy efficiency changes with increasing load. The power consumption caused by the data traffic cannot be measured and is therefore estimated. The size of the data sent is measured for one user and thus also for a sorted list. In contrast to the first scenario, the execution of the requests is not limited, which is why a high utilization of the hardware is expected. Here, a constant electrical power is expected again, since the same load is maintained by several requests over a longer period of time.

These two implementations mean that no external load driver is required in the test setup, since workloads are generated directly by the application itself. The selection and start of the respective scenarios are done by clicks in the user interface. This ensures that the process follows the example of a real web application, in which load is generated by user actions. However, the use of an automation tool is dispensable. This is because interactions to set the configuration are not part of the measurements and workloads are executed in an automatized way after the scenario is started. This also excludes that further software for automation influences the energy demand of the SUT. Furthermore, it is assumed that the user interface does not contribute to the dynamic consumption of the system, since it is not part of the investigations and does not differ in the different configurations. Additionally, the selection of workloads is chosen to maintain the expected constant electrical power for several seconds to ensure that measured values are not affected by system-induced fluctuations.

4.2 Specifications

In order to follow v. Kistowski's characteristics [15, 16] during the investigation, specifications of the SUT, frameworks used as well as the measuring device are listed in the following chapter. Since the selected specifications should always represent

scenarios as realistic as possible, all the points mentioned in the following explanations were selected for the measurement of a content management system in the form of a web application.

Hardware

The specifications shown below cover all the hardware used. This includes detailed information of both the SUT and the measuring device used. Since the selection of the measuring device has a significant influence on the selected SUT, this is described first in the following, before details of the computer system used are given.

To ensure that the measured values are as accurate as possible, an external current measuring device is used for the measurement. This is a GUDE Expert Power Control 1202-1, which was selected because it offers a low error tolerance. It also contains an interface for exporting the measured values. The readout of the data is done by a Python tool. Information about the current, the voltage and the current power factor are retrieved twice per second via SNMP. These values are then used to determine the electrical power.

By multiplying this by the time required for each calculation, the electrical work can be determined, which equals the energy consumed during this period.

In addition, no fixed installation of the device is necessary, which means that it can be used for further investigations afterwards. The only limitation of this model is that only values of one device can be measured. If several devices are connected at the same time, their aggregated power consumption is determined accordingly. Thus, no direct differentiation of the consumption of frontend and backend is possible. However, this fact can be compensated by adjustments in the measurement process.

Frameworks

In order to ensure the relevance of the results for the practice, it must be remembered that frameworks are often used in software development processes. Although they offer many advantages during the development process, there is no guarantee that they can also be operated in an energy-efficient manner. In order to be able to put the knowledge gained about changes in dynamic energy requirements in relation to practical consumption, it makes sense to use several frameworks for the creation of the application, which are also frequently used in current development processes.

Here, one framework is used for the development of the user interface, and one for the backend development. Their use also runs with JavaScript and C# in programming languages that are representative for the respective area.

For the frontend, React is the obvious choice. This is a JavaScript library, which allows a hierarchical structure of components of the user interface. These components can also perform various calculations using JavaScript. The resulting single-page web application also provides a good example of the structure of a modern web application.

A typical framework for server-side development is found in ASP.NET. This is a platform developed by Microsoft, which is specifically designed for the development of web applications. This part of the application is thus responsible for all server-side

calculations, as well as responding to the generated load. No server side caching was enabled in this experiment.

4.3 Setup of the Experiment

The experimental setup is one of the most crucial factors in energy efficiency measurements. The most commonly used approach is the System Under Test. A typical test setup includes the SUT, a power meter, a data acquisition and analysis system, and a load driver. From this, an energy efficiency report can be generated, which contains all important measured values of the test.

The selection of the computer system used and its exact hardware specifications can be made almost freely for measurements of the power consumption of software. It should only be noted that the level of static power consumption is directly related to the selected system. In order to make also smallest changes in the dynamic power consumption visible, it is helpful to use a system that is as small as possible, but can be representative for a practical application of the software.

Due to the restrictions caused by the selection of the measuring device, it is also not possible to separate the consumption of simultaneously acting clients and the corresponding server, since only their aggregated power consumption can be measured. For this reason, both the frontend and the backend can be located together on one system. This massively reduces the measured static power consumption, since only one device has to be operated. A commercially available laptop is used for the tests. Compared to desktop computers, the use of such a laptop has the advantage that notebooks are usually designed for mobile work and can therefore be operated in a more energy-efficient manner. Thus, results contain less static consumption, which makes even small differences in dynamic consumption noticeable. Specifically, this is a Lenovo ThinkPad X240 with 4 GB of working memory and an Intel Core i5-4300 Pro processor with a clock rate of 1.9 GHz. Devices like this are often used in office environments, which is why this model fulfills the condition of a practical system. In order to be able to determine the exact current consumption of the device, it must be ensured that any required energy is also recorded by the measuring device. For this reason, both batteries of the notebook were removed, allowing it to run exclusively in mains operation.

By implementing the two scenarios explained in the previous section, no external load driver is needed in the test setup, since workloads are generated directly by the application itself. The selection and start of the respective scenarios is done by clicks in the user interface. This ensures that the process follows the example of a real web application in which load is generated by user actions. However, the use of an automation tool can be dispensed with, since interactions for setting the configuration are not part of the measurements and workloads are executed automatically after the start of the scenario.

4.4 Experimental Procedure

In order to ensure the usability as well as the reproducibility of the obtained measured values, the execution of the experiment is of great importance. The used structure consists of three phases. First, the basic load of the system without software is determined. This is then repeated with software such as a web browser, before the application is tested under load in the respective configuration in the last phase. In both of these phases, it is expected that a constant energy consumption can be determined. However, as this is always subject to fluctuations in reality, measurements are taken over several minutes in these phases and the mean value of this period is calculated.

During the execution of the load, the average electrical power required for the period of the calculations as well as the time required for the sorting of a list is first determined independently of the scenario. From this, the electrical work performed per sorted list can now be calculated. By subtracting the electrical work performed during this period for the basic consumption, the additional consumption per sorted list can be calculated.

For server-side calculations, however, this does not yet include all the consumption that occurs. Thus, by applying this calculation method, only the electrical work of the backend is obtained. However, this scenario also results in a non-negligible additional consumption due to additional data transmission on the Internet. Similar to the front-end load, this is determined once for the respective list size and multiplied by the number of desired users. However, since this value can only be estimated, the value determined by Aslan is used in the following chapter. In addition, it is assumed that his observation from 2015 of a 2-year halving of the energy demand continues to correspond to the real development. For this reason, an estimate of this value in the following studies is 0.0075 kWh/GB. This corresponds to 0.027 Ws/kB.

This procedure now makes it possible to directly compare the power consumption of both implemented configurations.

5 Results and Discussion

First of all, in the measurements carried out, it turned out that the basic consumption of the system was not measurably affected by the started application in idle mode. Only the web browser slightly increased the required electrical power. Thus, a uniform basic consumption of 4.9 W applies to both scenarios.

After performing several series of measurements using different lists with multiple repetitions in both scenarios, it was possible to determine average values of the electrical power, from which the final results of the investigation are derived. A comparison of the most important measured variables of both configurations is shown below in Table 1.

It can be seen that the electrical work consumed per sorted list differs greatly between the two scenarios. The configuration that carries loads on the client side

Table 1 Comparison of the average results for one list each. Scenario 1 includes client side and scenario 2 server side computations

Measured variables	Scenario 1	Scenario 2
No-load electrical power (W)	4.9	4.9
Average electrical power under load (W)	6.22	21.28
Time (s)	0.011	0.001
Gross utilization (Ws)	0.068	0.021
Additional consumption (without network) (Ws)	0.021	0.016
Amount of data sent (kB)	0	47.9
Consumption in network (Ws)	0	1.29
Total additional consumption (Ws)	0.021	1.306

has a significantly lower energy requirement. Nevertheless, it also shows that these differences are largely due to the data transfer in the network. The sole calculations, on the other hand, can be executed more economically on the server side.

In the following, these results will be applied to the research questions defined at the beginning. For a better understanding, this detailed consideration is carried out separately for the individual focal points of the investigation. In addition, recommendations for action are given for energy-saving load distribution within a web application.

The first and thus fundamental research question of this paper is "How strongly does load balancing influence the energy efficiency of a web application?" This question can be answered directly from the collected data. Thus, it could be clearly shown that different implementations, which differ only in their load distribution but not in the calculations performed, show significant differences in energy efficiency, i.e. the amount of energy consumed for the same work. Increases in the consumption of the entire system beyond 62 times were found. In addition, savings of about 30% were demonstrated for the efficiency of the calculations alone. It was found that these savings are not limited to a specific scenario, but areas of potential savings are present in both scenarios. This becomes particularly clear in the direct comparison of both scenarios carried out. Although savings could be achieved in scenario 2 during the calculations, these were massively over-compensated by additional data traffic. For these reasons, it can be clearly concluded that the load distribution has a very strong influence on the energy efficiency of a web application. However, it should be noted that differences in load distribution result in savings in different areas.

The next step of the investigation deals with the question "Which factors are causal for the differences that arise?" Through the investigation, two main factors could be identified by which the energy efficiency of web applications is mainly influenced.

Network transmissions form the first determining and most important factor. It could be shown that even under the assumption of a 2-yearly halving of the consumption arising thereby it can come to a clear overcompensation of savings, if additional large data quantities must be sent. It should be noted that the calculated values are

only based on estimates. The consumption assumed in the calculations was therefore 0.027 Ws/kB. This is because the exact consumption in the network in real use can be influenced by various factors and is therefore not measurable. Nevertheless, the calculated values offer a point of reference and represent energy consumption through network transmissions as a clearly determining factor for the energy efficiency of a web application. It is worth mentioning here that only data transfers that occur additionally due to load distribution should be taken into account.

The energy consumption during the calculations provides a second point of contact for savings. Thus, with the help of the measured values, it could be proven that differences of 30% in the energy demand for handling the workloads occur depending on the calculation location. Here, however, a further subdivision must take place, since several points exist that influence this factor.

According to the study of Grosskops, the extent of **utilization of the system**, which accomplishes the computations, has a relevant influence on the power consumption, which results for work expenditures [19]. The way the application was implemented made it possible to include this factor to a satisfactory degree. In scenario 2, for example, simultaneous user requests resulted in a significantly higher workload than in scenario 1. This initially manifested itself in a noticeably higher average demand for electrical power. Nevertheless, this increase results in a significant reduction in processing time, which reduces the electrical work performed per workload.

In addition, the **complexity of the load** to be handled must also be taken into account. Thus, it can be assumed that the size of the possible reductions in energy demand is directly related to the time required for the respective work effort. It is therefore obvious that the more complex the calculation to be performed, the greater the expected savings per work performed. With the help of the developed application, the same calculations as those used in the previous chapter could be performed with a shortened list. In the course of these executions, this connection was likewise recognizable.

Furthermore, the influence of the respective **programming language** may not be neglected. Thus, investigations of different programming languages could already determine that partly large differences exist between them. For this reason, JavaScript in the frontend and C# in the backend were used as common programming languages for the respective area, which is why it can be assumed that the values determined already include this factor. Nevertheless, other combinations of programming languages or frameworks are also conceivable. Against this background, these should always be selected and considered depending on the requirements of the software.

However, the extent to which all of the factors influence the energy efficiency of calculations and how great their effect is on the values determined in this work was not investigated in this paper. For this reason, no further weighting of these factors can take place. However, it is obvious that all mentioned points have contributed to the determined differences.

The last research question, “How does an optimal load distribution for web applications result in practical application scenarios?” follows from the first two questions, since all relevant factors must be included for this. In addition, there are some

limitations to answer this question, which are listed below. For example, not every workload can be executed freely interchangeably at the client or on the server. This may, among others, be due to security concerns, data protection, or even secret source code. In addition, particularly computationally intensive processes can often not be outsourced in a practical way on the client side. Another limitation arises from the choice of technology. For example, selected frameworks partially limit the complexity of logic that can be executed on the client side. All of these reasons are not considered in the following. Instead, we will only talk about computational efforts that can be executed on the client side as well as on the server side.

For the case of sorting lists, based on the measured values obtained it can be concluded that workloads of this type should be handled on the client side if the goal of an energy-saving web application is pursued. This is based on the fact that the simulated workloads are small and can be handled very quickly, while a larger amount of data must also be sent. However, this is only one of many conceivable use cases and thus only a small excerpt from practice. Nevertheless, the knowledge gained about the determining factors and their respective influence can be used to draw conclusions about practical applications. Due to the manifold conceivable application scenarios, however, this is only possible in the form of tendencies.

As shown by the previous research question, the energy demand for network transmissions is the determining factor. Although the additional consumption resulting from this is only an estimate, it was shown that these potential savings are clearly overcompensated by calculations that are more efficient. From this, the recommendation can be derived that calculations for which additional large amounts of data have to be transported should ideally always be performed at the point where the data to be processed is already available. However, it is important to note that only additional data consumption and thus the data traffic, which would be avoided by calculations on the client side in contrast to server-side executions, is considered. In line with this, scenario 1 turned out to be more energy-efficient.

For use cases in which such data traffic does not occur or only occurs in small quantities or where data transmission cannot be avoided, further considerations must be made. In this case, it can be assumed that calculations in the backend require less energy than in the frontend, taking into account all the factors mentioned. Unfortunately, based on these results alone, no statement can be made about the complexity at which energy savings for calculations compensate for the additional consumption in the network. This opens the need for special investigations for the respective use case. Nevertheless, a general tendency is recognizable. Although the measurements carried out showed savings of over 30%, in absolute terms this only represents a saved electrical power of 0.005 Ws per list. This reduction is minimal in relation to, for example, 4.3 Ws, just for operating the laptop for one second. Furthermore, it must also be considered that these savings occur per workload and thus also per user of the application. Furthermore, if all the factors shown so far exert the greatest possible influence, the savings achieved in this area can increase even further. This is especially true for more complex calculations, since the sorting of lists can be performed with comparatively little computational effort.

6 Summary and Outlook

In summary, it can be stated that calculations for which a large amount of data has to be transmitted can only be performed in a particularly energy-efficient manner if this data transmission is as minimal as practically possible. Whereas loads with only a small amount of data but nevertheless high complexity and high time requirements should always be executed on the server. Another practical possibility is to divide the load between the client and the server. In certain scenarios, preprocessing and thus a reduction of the amount of data could take place on the client side. Based on this, calculations that are more complex can then be performed by the server. This approach would achieve a reduction of the amount of data submitted along with a more efficient calculation. Based on these findings, it should also be noted that an energy-saving web application can reduce consumption most effectively by minimizing data transmission. Since this is not possible only based on the load distribution, this should represent a criterion in all ranges of the software development. Starting points for this can be found, for example, in studies by Kern, in which specific measures are presented [23]. Similar starting points also already exist in other publications [24].

References

1. Hilty, L.M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., Wäger, P.A.: The relevance of information and communication technologies for environmental sustainability—A prospective simulation study (2006)
2. Molla, A.: GITAM: a model for the adoption of green IT. In: ACIS 2008 Proceedings, pp. 658–668 (2008)
3. Gröger, J., Köhler, A., Naumann, S., Filler, A., Guldner, A., Kern, E., Hilty, L.M., Maksimov, Y.: Entwicklung und Anwendung von Bewertungsgrundlagen für ressourceneffiziente Software unter Berücksichtigung bestehender Methodik - Abschlussbericht (2018)
4. Morley, J., Widdicks, K., Hazas, M.: Digitalisation, energy and data demand: the impact of Internet traffic on overall and peak electricity consumption. Energy Res. Soc. Sci. 128–137 (2018)
5. Stobbe, L., Proske, M., Zedel, H., Hintemann, R., Clausen, J., Beucker, S.: Entwicklung des IKT-bedingten Strombedarfs in Deutschland. Bundesministerium für Wirtschaft, Berlin (2015)
6. Bundensetzagentur, Tätigkeitsbericht Telekommunikation 2016/2017 (2017)
7. Hilty, L.M., Lohmann, W., Behrendt, S., Evers-Wölk, M., Fichter, K., Hintemann, R.: Grüne software: Ermittlung und Erschließung von Umweltschutzpotenzialen der Informations- und Kommunikationstechnik (Green IT) (2015)
8. Philippot, O., Anglade, A., Leboucq, T.: Characterization of the energy consumption of websites: impact of website implementation on resource consumption. In: 2nd International Conference on ICT for Sustainability, pp. 171–178 (2014)
9. Hindle, A.: Green mining: a methodology of relating software change to power consumption. In: 2012 9th IEEE Working Conference 2012, pp. 78–87 (2016)
10. Singh, J., Naik, K., Mahinthan, V.: Impact of developer choices on energy consumption of software on servers. Procedia Comput. Sci. 385–394 (2015)
11. Vasques, T.L., Moura, P., de Almeida, A.: A review on energy efficiency and demand response with focus on small and medium data centers. Energy Effic. 1399–1428 (2019)

12. Acar, H.: Software development methodology in a Green IT environment. Université de Lyon (2017)
13. Aslan, J., Mayers, K., Koomey, J.G., France, C.: Electricity intensity of internet data transmission: untangling the estimates. J. Ind. Ecol. **22**(4), 785–798 (2018)
14. Dick M., Naumann S., Held A.: Green web engineering—A set of principles to support the development and operation of "Green" websites and their utilization during a website's life cycle. In: Proceedings of the 6th International 2010, pp. 48–55 (2010)
15. Kistowski J., Arnold J.A., Huppler K., Lange K.-D., Henning J.L., Cao P.: How to build a benchmark. In: Proceedings of the 6th ACM/SPEC, pp. 333–336 (2015)
16. Kistowski, J., Lange, K.-D., Arnold, J.A., Sharma, S., Pais, J., Block, H.: Measuring and benchmarking power consumption and energy efficiency. In: Companion of the 2018 ACM/SPEC, pp. 57–65 (2018)
17. Pinto G., Castor F., Liu Y.D.: Mining questions about software energy consumption. In: Proceedings of the 11th Working, pp. 22–31 (2014)
18. Johann T., Dick M., Naumann S., Kern E.: How to measure energy-efficiency of software: metrics and measurement results. In: First International Workshop 2012, pp. 51–54 (2012)
19. Grosskop, K.: PUE for end users—Are you interested in more than bread toasting?. In: 2nd Workshop EASED@BUIS 2013 Energy, pp. 15–16 (2013)
20. Pang, C., Hindle, A., Adams, B., Hassan, A.E.: What do programmers know about software energy consumption? IEEE Softw. **33**(3), 83–89 (2016)
21. Torre, D., Procaccianti, G., Fucci, D., Lutovac, S., Scanniello, G.: On the presence of green and sustainable software engineering in higher education curricula. In: 2017 IEEE/ACM 1st International Workshop, pp. 54–60 (2017)
22. Kern, E.: Green computing, green software, and its characteristics: awareness, rating, challenges. In: From Science to Society, pp. 263–273 (2018)
23. Kern, E.: Nutzerzentriertes green web engineering. In: GI Edition Proceedings 220—Informatik 2013—Informatik angepasst an Mensch, Organisation und Umwelt, pp. 966–975 (2013)
24. Dick, M., Kern, E., Johann, T., Naumann, S., Gülden, C.: Green web engineering—Measurements and findings. In: Man—Environment—Bauhaus. Light Up the Ideas of Environmental Informatics, pp. 599–606 (2012)