TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Green coding

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Contents

Green Programming

Panu Puhtila - Reading materials
Oshani Weerakoon - Exercises, lecture videos and examples

1 Introduction

As the spectres of global warming and climate change loom over the horizon of tomorrow, achieving sustainability and reducing the environmental impact of the software development has become more relevant, and is rapidly transforming the landscape of the IT-sector. When in the past optimization of the algorithms was done purely to enhance performance, now it is sought for the minimization of the carbon footprint. Green programming, or green software development, as it is sometimes called, is a design philosophy that has arisen to answer these issues, emphasizing the need to reduce the energy consumption, and thus the carbon foot- and handprints, produced by the ICT -industry.

During the last decade, as both mobile devices and the internet of things, and with them the use of internet has become widespread across the global population, the energy demands of the ICT -devices have expanded to a point where the 2017 study [10] estimated that back then 4% of all energy consumption globally, and 2% of all carbon emissions were the result of ICT-devices. The share of the global energy consumption of the ICT-devices is expected to rise to at least to 21%, and perhaps even as high as 51% by the 2030. Other studies estimate that the energy consumption of the ICT-sector would double every three years, for at least the next two decades from now onwards. [29]

Several studies have shown that the composition of the deployed software affects the energy consumption significantly, although the exact effect varies between the differing hardware platforms. In server side it has been estimated that the correct optimization and design of the software can cut down the energy consumption by almost 40%[35], and on client side roughly 20%[27]. Other studies have shown that there is a real demand for further education in these matters [24], as the majority of software developers don't have a clear idea of what constitutes to the energy consumption of the software, or how to reduce it.

1.1 What is green programming?

In practice green programming and software development means several things:

- *Design of better and more efficient algorithms*

- *Measurement of energy consumption during the development process and/or during the application run-time*
- *Design of the software development process with the aim of achieving lower carbon footprint*

Design of better and less resource-intensive algorithms is at the heart of green programming, as it is undoubtedly the most direct way of affecting the memory usage, and thus the energy consumption of the application. The specifics of how exactly to design better algorithms will be discussed later during this course.

Measuring the energy consumption of the application can take two forms; it may happen either during the development process, in which case we are talking about the so-called "energy-aware developer", or it can happen during the run-time of the application, in which case we are talking about the so-called "energy-aware application". In the first case, the development team uses both software- and hardware -based means of observing the energy consumption of the application, using this information to guide the development process. In the latter case, the application is designed so that it has several different "energy-consumption profiles", and it can automatically adjust it's own performance to achieve lower energy consumption as desired.[1] The latter approach has been adopted somewhat in the mobile application development, to minimize the battery drain.

The last point in the list involves questions such as what platforms and frameworks to use, what programming languages the application will be written in, what cloud services to employ etc. In other words, what organizational-level development choices must, or at least should, be made to achieve the wanted level of energy consumption. Usually these kinds of decisions are not for the rank-and-file -developers to worry about, and extend beyond the choices in IT-technologies to be used to other practical concerns such as carbon-footprints of the work commuting, environmental impact of having permanent offices and so on. These aspects of the development process will be further discussed in the subsection 1.6.

1.2 Carbon footprint: what is it and how is it measured?

Since the reduction of carbon footprint is central to the whole idea of green software development, we shall take a look at how exactly the carbon footprint is defined and how is it measured in practice. Commonly accepted definition for the carbon footprint is:

A measure of the total amount of carbon dioxide (CO2) and methane (CH4) emissions of a defined population, system or activity, considering all relevant sources, sinks and storage within the spatial and temporal boundary of the population, system or activity of interest. Calculated as carbon dioxide equivalent using the relevant 100-year global warming potential (GWP100).[19]

Several methodologies of calculating the carbon footprint of products and processes exist, for example the Life-Cycle Assessment (LCA) [13] and Greenhouse Gas Protocol (GHG) [28]. The LCA is used mostly in evaluating singular processes and products, while the GHG is mainly intended for the scope of countries and large corporations. While sometimes only the CO_2 is considered in the calculation of the carbon footprint, hence the name, it is more common to use methods of calculating the carbon dioxide equivalency for other types of greenhouse gases also and include these to the total carbon footprint. For example the GHG is used in this fashion. However, influential climate authorities such as IPCC (International Panel for Climate Change) use a method that counts only the carbon dioxide emissions [33], justifying this approach by claiming that the climate impacts of the other greenhouse gas emissions are harder to quantify. This approach has resulted in some criticism from other climate scientists, as omitting other greenhouse gases from the equation might make certain forms of industry look more greener than they actually are.

In theory, the measurement of the carbon footprint by the LCA is conducted in the following manner:

- **Goal and scope**: *Declaration of the definite goal of the study, and exact specification of the scope it is conducted in*
- **Life Cycle Inventory**: *Compiling an inventory of relevant energy and material inputs and environmental releases*
- **Life Cycle Impact Assessment**: *Evaluating the potential environmental impacts associated with identified inputs and releases*
- **Life Cycle Interpretation**: *Interpreting the results to help you make a more informed decision*

As can be seen from the above list outlining the phases of the procedure, conducting the LCA is very much akin to a scientific research project, rather than using a ready-made tool to measure the effect, and it requires a certain degree of specific scientific expertise from those who use the method. As should be obvious from the steps outlined above, the practical application of this kind of investigation can take many forms. Due to it's nature and the sheer complexity of the procedure the LCA has been criticized for being somewhat prone to the bias of those who conduct the survey, as it has been observed that there can be great variation in the results.[31]

Other widely used tool, or rather a set of tools, is the Greenhouse Gas Protocol, which is globally used by the majority of important industrial actors. GHG is a collection of standards and measuring instruments which are meant to facilitate the tracking of greenhouse gas emissions, which are designed for such a scale that they are mostly used by aforementioned industrial giants and entire nations.[1]

[1] Greenhouse Gas Protocol

1.3 Carbon footprint and environmental impact across the life-cycle of the software

As the LCA is originally designed for assessing the environmental impact of the physical products, its' traditional implementations are not straight-out-of-the-box -suitable for analysing the environmental impact of the software development process or the software life-cycle. The main reason for this is that software development does not include things like extraction, transportation or storage of raw resources which are then refined to products, but is more akin to other immaterial production processes such as writing or animation, where the final product is "constructed" from "parts" that never existed in material form. Dick et al. [9] addressed this issue in 2010, when they released a paper outlining a method for adjusting the LCA for the needs of the software development. In their model, the sustainability assessment of the life-cycle of the software is broken down to following phases:

- **Development**: The environmental impact of the development process in itself is considered in this phase.
- **Distribution**: In this phase the impacts from the distribution of the software, nowadays mainly the maintenance of the online download -repositories and services, size of the downloaded files and the resulting network workload is taken in to account.
- **Acquisition**: The impact that the acquisition of the software by the customers has on environment. This phase is somewhat obsolete nowadays, as most software purchases are made online.
- **Deployment**: In this phase the impacts of deploying the software are taken in to account. The main environmental questions are the hardware requirements of the software, and certain logistical considerations in the case of larger organizations.
- **Usage**: Impacts from the usage of the software, such as run-time network load and memory usage are considered under this phase.
- **Maintenance**: Update frequency, update sizes and the upkeep of the organization that is responsible for these and the customer service of the software are the main considerations in this phase.
- **Deactivation**: Main considerations in this phase are the impacts of the long-term backups and the conversion of data for future use.
- **Disposal**: Since very few contemporary software products use physical mediums, manuals or packaging anymore, this phase is mostly obsolete. If, however, the software would use some of the aforementioned physical trappings, the impact of their disposal should be considered here.

While the nature of information and communications technology has changed somewhat from the 2010, with physical mediums for software distribution becoming almost obsolete and online acquisition of software becoming almost universal, this doesn't change the fact that the model described above is still useful framework through which to classify the various aspects of the environmental impact of the software development.

1.4 Computer components from the perspective of energy consumption

While this course is about software development, and specifically about programming, since the energy consumption is something that ultimately happens due to processes running in the hardware of the computer, we felt that it would be useful to take a brief glimpse on the fundamental parts of the computer architecture and what is their (usual) role in the aforementioned consumption. In the figure 1 you can see a rough overview of the central components that any modern computer has.

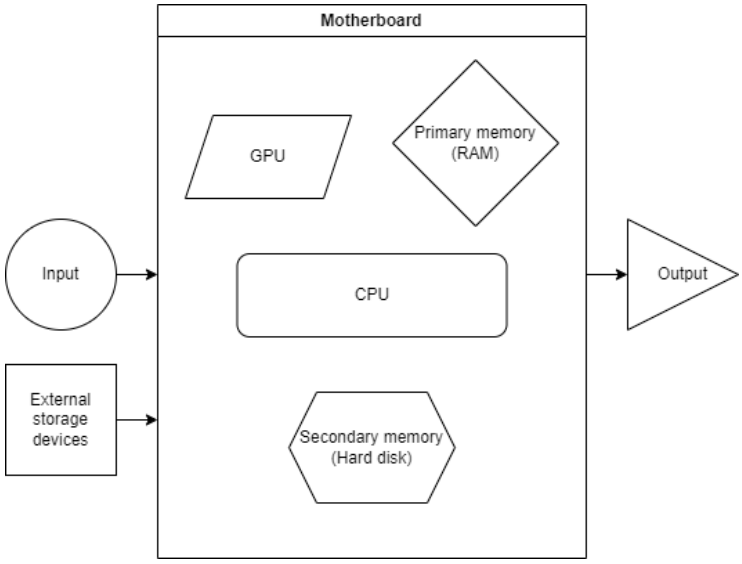

Figure 1: Overview of computer components.

The core of every computer system is the Central Processing Unit, or CPU. It's the muscle beneath the iron frame that crunches the numbers and does the lifting, so to speak. For this reason it's also usually, but not always, the part of the computer where the majority of the energy consuming activities happen. The power consumption of a processor consists of both static and dynamic components. By static here is meant the constant standby-level of power required to power the processor and keep it ready to execute tasks, and

by dynamic the tasks that the processor executes.[7] Some estimates [36] put the energy consumption of the processor at roughly 25% of the total used by the computer.

While in the past the CPU was the only muscle the computers had, to answer the specialized needs of the modern-day computing another heavy-lifting component was evolved. Graphics Processing Unit, or the GPU, is akin to CPU in such a way that it too crunches the numbers and does the lifting, but unlike the CPU which is more-or-less a multi-purpose tool, GPU is a device specifically intended for doing certain kinds of mathematical operations. As it's name implies, it was originally intended for the processing of rendering tasks required for modern-day computer graphics, but in the past two decades other applications for it's special nature have been discovered, such as the processing of block-chains. While the newest generation of cryptocurrency -mining devices have moved onwards from the use of GPU's to ASICs (Application Specific Integrated Circuit), for the past decade the attributes of GPU's led to a development of specialized cryptocurrency-mining architectures which were essentially just stacks of GPU's wired together to process the block-chains for the production of cryptocurrencies such as Bitcoin and Ethereum. As a sidenote, it has been estimated that the annual global Bitcoin -production alone consumes energy slightly more than the entire country of Norway per year[2]. However, certain studies on green software development have resulted in to recommendations that software utilizing GPU should be used more, as while the GPU in itself uses more energy than the CPU, it's also much more energy-efficient.[23]

Third big energy consumer in the computer architecture is the memory, which according to Vogelsang [36] uses up roughly 20% of the total energy consumed by the machine. Memory comes in two types: The run-time, or primary, memory known as RAM (Random Access Memory); and the secondary, or storage memory, usually known as hard disk space. RAM, together with CPU, is the only absolutely required part of any computer architecture; everything else apart from these two components is more-or-less optional, depending on the wanted usage of the computer. In stand-by mode, it has been estimated that the RAM consumes roughly 1/5 of the power used by the CPU, but when performing RAM -intensive operations this consumption can rise to be equivalent with the consumption of the CPU.[21] Even somewhat modern and thus relevant estimations about the hard-disk power consumption are sadly quite scarce, but from the studies available it can be seen that the newer SSD -drives use roughly 50% of the energy in read-operations, and roughly 75% of the energy in write-operations, relative to the older HDD -drives.[15] It should also be noted that while idle, power consumption of the SSD -drive is very low.

The external devices of the computer, i.e. inputs, outputs and different kinds of external storage devices and special purpose equipment usually don't contribute much to the energy consumption. In the past, the displays of the computers used almost as much energy as the CPU, but with the advancements in the LED- and LCD-technologies this has fallen drastically. There are some

[2] Annual energy consumption of Bitcoin

special external devices which may use more energy than the common externals, but such instruments are quite rarely used by the ordinary consumers.

1.5 What affects energy consumption?

Since the energy consumption, and it's reduction, are the key issues in green programming, we must take a look at what exactly constitutes to increase the consumption. However, it turns out that this question is not exactly simple one to answer, as it is quite hard to exactly pinpoint which processes are more energy-intensive and which are not. In addition, the energy consumption of the same software varies between the hardware platforms it is used on, which makes the issue even more complicated.[14] Thirdly, there is evidence that the choice in programming language affects the energy efficiency of the software that is written. [5]

In essence, there are two ways to approach the issue: Either through principles, models and "rules-of-thumb" based on observations made during research on energy consumption, or by software solutions and hardware devices that can pinpoint the energy consumption in specific parts of the target application.

1.5.1 Principles of energy consumption

While it is common conception that the engagement of the CPU, i.e. the processing power required, correlates with the energy consumption of the application, this is only partially correct, since not all processes are CPU -bound. Some processes are IO -bound, others event -bound, and some are run on peripherals, i.e. GPU, in which case their engagement with the CPU is quite minimal, and thus the CPU -requirements tell little about the energy consumption of these applications. On processes that are CPU-intensive, there is in general a strong correlation between the used processing power and the energy consumed in doing so [14]. However, it has been proven that this is not always the case either [7], and thus it should be kept in mind that the engagement -level of the processor is not necessarily a reliable indicator for the energy consumption.

As another rule-of-thumb, certain researchers have argued that the amount of system calls the application makes can be used as a relative indicator of the energy consumption, as there exists a strong correlation between the amount of system calls and the consumed energy. Thus, it can be inferred that if the newer version of the software uses less system calls relative to it's previous versions, it will also use less energy.[26]

Lewis et al.[20] developed in 2008 a linear regression model, which relates the processor power, bus activity and ambient temperature of the system, and is capable of producing real-time predictions of the power consumption. The model is too complex to be explained in detail in the scope of this course, but it is presented here as an example of how to measure the power consumption with the combination of mathematical methods and some understanding of the relevant processor architecture.

Residing somewhere between the insight born from research data and a software meant to measure energy consumption is the GreenOracle, a machine learning model developed by Chowdhury and Hindle[4], which can predict the energy consumption of the software application in joules with roughly 90% accuracy.

1.5.2 Measuring software and devices

Several different tools exist for the developers to use in measuring the energy consumption during the development process, ranging from physical devices to measurement applications. These will be discussed at greater length and detail in the Section 4, so we will provide here only a very brief overview of the subject.

Physical devices that are used to measure the energy consumption are essentially the same tools which are used for measuring the attributes and usage of electricity in other technological disciplines. The most common and accessible of these are **AC -power meters** that are nowadays quite easily obtainable from home electronic appliance -stores, due to them being used together with the smart-house devices. However, depending on the application and the wanted level of measurement detail, this kind of meter might not be sufficient. If greater detail or some specific functionality is needed, the potential measurement devices range from **bench power supplies** to **power measuring integrated circuits**. Depending on the specific device these solutions are often either quite expensive, very specific in application and/or require profound understanding of the theoretical aspects of the power measurement to be calibrated correctly.[3] Thus, their suitability and feasibility for the needs of a specific software development project must be evaluated more carefully than with more general measurement instruments.

Software that is used for measuring the energy consumption comes in many forms, from device-manufacturer provided applications to open-source projects. The former are usually device- or manufacturer -specific, while the latter are usually built around some functionality exposed by the device-manufacturers, such as Intel RAPL (Running Average Power Limit), a feature in the post-Sandy Bridge -microarchitecture Intel -processors which allows for the monitoring of the power consumption happening in the processor through custom-built applications.[17] Some power measuring applications are freely available online, such as Green Algorithms[4], a carbon footprint -calculator developed by Lannelongue et al. in 2021. [18]

1.6 Sustainability in development process

Minimizing the carbon footprint in green software development is not limited only to the changes in the actual programming practices, but includes also decisions and considerations that happen in the more managerial layer of the development process. Decisions such as whether the developers should commute

[3] Information on measuring devices in Finnish
[4] Green Algorithms

to work every day or work remotely, whether to meet the stakeholders face-to-face or teleconference with them, whether to maintain a permanent office space and so on are usually things which the rank-and-file -developers do not need to worry about, but which still affects the energy consumption considerably, and for this reason we will provide a brief overview on the current consensus on the subject.

Kern et al.[16] estimated in their 2015 study that commuting to work every day increases the carbon footprint of the development process by as much as 24%. In their calculation maintenance and heating of the office space for the duration of any given project accounts for approximately 80% of the total carbon footprint of the project. It should be noted that their estimations are generated by the model they developed for calculating carbon footprint of the software development process, and the example results obtained are based on modellings made on micro-enterprises (less than 10 employees) operating in Germany, and might thus not be representative of the larger global situation. However, these results are indicative of what kind of effect remote work or switching to renewable energy, if adopted more widely, could have on carbon footprint.

Gupta et al. [12] conducted a slightly larger scale investigation on the carbon footprint of the ICT -industry, using the aforementioned Greenhouse Gas Protocol. In their analysis the relative impact of the software industry operations was negligible compared to the effect of hardware manufacturing, which was deemed to be the most environmentally polluting side of the whole computing business. This is mostly due to the emergence of green and renewable energy options, which have led to many ICT -industry operations becoming essentially carbon neutral.

Of course, it should be understood that the paradox of green energy is always present here; if a given operation is powered completely by renewable energy, it means that some other operation must be powered with non-renewable energy, as there simply is not enough sources of renewable energy for everyone. Thus the idea of fixing the carbon footprint -problem by switching to renewable energy is efficient only when all energy production has become renewable, which is long way in to the future. Until then, fixing the environmental impact by switching to green energy, but not seeking to decrease the overall energy consumption, is akin to moving money from pocket to pocket and claiming that it is profitable. Of course, the production of renewable energy is constantly increasing[5], but for the immediate future it will not be the majority of global energy production, and thus can't be solely counted on to solve the issues of carbon footprints.

[5] Renewable energy production

2 Green Programming Practices

2.1 Selecting the tools for green development

2.1.1 Programming languages

The choice of the programming language is surprisingly important from the viewpoint of the green programming, as research has shown that different programming languages vary quite significantly in their energy-efficiency. There is, in general, a difference between the compiled and the interpreted languages, with the compiled ones usually being less energy consuming and better performing than the interpreted ones. In line with this, languages such as Java, which are interpreted and then compiled just-in-time are more efficient than purely interpreted, as the JIT can salvage some of the performance that is lost in the interpretation process.[6]

Couto et al. published in 2017 a study [5] where they compared 10 popular programming languages in various tasks, and established a "Green ranking" for them based on their results. In general their results indicated that faster processing languages tend to consume less energy than slower ones, although there are specific cases where the slower languages are more energy-efficient. To give meaningful context to frame the question of how the energy-efficiency of these ten languages was compared, we must explain the test sequence used by the Couto et al. Their study was based on an earlier research framework called Computer Language Benchmark Game (CLBG), which is a set of 13 computational problems, each of which is implemented in 27 different programming languages. Each of the solutions in the CLBG are developed by verified professional experts in using these languages, and thus represent the state of the art on how to best and most efficiently and elegantly solve these problems. From this CLBG -set Couto et al. chose ten programming languages, which were C, C#, Fortran, Go, Java, JRuby, Lua, OCaml, Perl, and Racket, and their implementations for ten different problems, and then developed an energy-measurement application with C and Intel RAPL to determine how much energy each of the solutions consumed.

In short, the C was by far the least energy-consuming language of those surveyed, taking the first place in 7 out of 10 problems. In general both Perl and JRuby were the most energy consuming ones. Java, OCaml, Go and Fortran competed with the C for efficiency, but apart from three studied scenarios where C did not prevail they were slightly less energy-efficient. In the majority of tests the rest of the studied languages fell somewhere between the Fortran and Perl in terms of energy-efficiency.

Since the language used in this course is JavaScript, and since it was not evaluated in the research explained earlier, we must take a look at the studies done on its' energy efficiency. As this kind of research stretches back over a decade and some of the older studies are done on hardware and software - environments that are no longer in use, we have focused here on the newer

[6] Video: MIT - Matrix Multiplication

research on energy consumption of JavaScript.

Van Hasselt et al. [34] conducted a comparative analysis on the differences between the energy efficiency of the JavaScript versus WebAssembly compiled from C, and came to conclusion that WebAssembly-C is by far less energy-consuming than the JavaScript. Considering the results obtained earlier by Couto et al [5], this is hardly surprising. Similar results were obtained by Macedo et al.[8], who concluded that WebAssembly-C is roughly 30% more energy-efficient than JavaScript. Both of these studies also noted that the choice of what browser was used in executing the code affected energy consumption greatly, with Google Chrome being the least energy-efficient browser in common use.

However, the reason we are using JavaScript in this course and not WebAssembly is because ultimately WebAssembly is not something you could solely use to create websites or mobile applications with. Nor is WebAssembly a language in itself, but rather a standard by which several different languages such as C, Rust and C++ can be compiled to run in browser.[7] WebAssembly is mostly good for doing "heavy lifting" in the client side, if it is needed, but whether you use it or not you still need to use JavaScript, CSS, HTML and all the other usual technologies for web development, so sadly it does not provide a green solution for the all of the issues of energy consumption as such. For some specific situations it provides such a solution, and if applicable it should be used, but for the foreseeable future JavaScript will still be the de facto standard in web application development, and thus we must focus on how to use it in the most energy efficient ways possible.

Thus, the choice of what programming language to use, when considering the environmental impact, is almost always a compromise of some kind. In theory, this compromise is dictated by the needs of the project one is engaging in, and differs from the usual considerations of what language to choose only in that the goals of green programming are kept in mind so that when the decisions are made between two otherwise equally suitable languages, the one which is less energy-consuming is chosen. However, as the examples and research data in this chapter demonstrate, the choice is rarely so clear-cut.

2.1.2 Frameworks & content-management platforms

Modern software development is heavily dependent on the use of both frameworks and content-management systems (CMS) to streamline the development process and standardize the production quality. While the frameworks and CMSes offer a plethora of ready-made tools and component libraries upon which to build your software, they also come with the burden of lots of unused code and infrastructure that just takes up space and consumes energy pointlessly.

The effect on the energy consumption from the use of software development frameworks has not received lot of attention from the researchers, but from the few studies conducted on this phenomenon we can make the conclusion that in

[7] What is WebAssembly?

general using the frameworks and external libraries, is detrimental in the sense of energy-efficiency. [3] From the perspective of green programming this result is quite negative, as not using the frameworks at all is simply not an option for the majority of the software development industry.

However, while frameworks and CMSes are useful, they are also often used needlessly. If you're going to make a small website with mostly static information, using ReactJS or Drupal for the job might be a bit too much, and just the time it takes to setup these kinds of environments for smaller tasks overshadows the time you would spend doing the task in the first place. Thus the necessity of using this or that framework or CMS should be evaluated on per-case -basis,

2.2 Unoptimized data handling

When we are talking about unoptimized data handling, we are essentially talking about the file-size -creep and -bloat that has been happening in tandem with the development of more efficient data storage and transfer technologies, and how to minimize this kind of unnecessary resource consumption. The past three decades have witnessed completely unprecedented increase in the technological capability to both store and transmit data, and as a side-effect, this has led to both the software file size growth and abundance of network traffic. While sometimes this increase in resource usage is justified, in many other cases its' purely the result of wasteful coding and management.

2.2.1 Unnecessary data transfers

- **Why:** Majority of the modern software transfers data either internally or externally, and while this is often necessary part of the functioning of the application, there is also lots of completely useless data transfer taking place. Usually the types of transferred information fall in the following two categories:

  1. **Data transfer between the parts of the application:**
     This is often defined by a protocol established between the parties of the data transfer.
  2. **Data transfer happening between the server and the client:**
     Such as contents of the website, streamed videos and attachment files transferred from client to server.

- **How:** Reducing this kind of data transfer capacity may be hard, as the established protocol affects the energy consumption significantly, but one avenue of reducing the consumption is by optimizing for example the communication between the application and the database, so that when the database is queried for information only necessary variables would be fetched, and not all the contents of the table row, if they are not needed.

  As a rule of thumb, unneeded data should never be transferred between the parts of the application.

2.2.2 Reducing data transfer quality

- **Why:** Large portion of the data transfer size comes from the content data, such as images and videos. Reducing their quality even slightly can have a large impact on the carbon footprint, while having no discernible reduction in the user experience. Other way of approaching the reduction of data quality is by adjusting the computational accuracy to level that is needed for the application to reach its' intended usage goal.

- **How:** Regardless of language or framework, this can be achieved through making such design choices that define for example how large image files are allowed to be inputted by the users, or what resolution is required from the geolocation data. Languages and frameworks often also offer either ready-made libraries which have the functionalities to for example control the image size through code, or such can be written as needed.

- **Example:**In this example, we will create a Node.js application that serves optimized images to the client.

2.2.3 Bundling

- **Why:** Considering the composition of the modern applications it should be noted that a large majority of them consist mostly of external libraries, out of which only several percents of the code is utilized, and the original source code of the application contributes only several percent of the total code of the application. For this reason bundler -applications should always be used when shipping the source code, so that the unused "extra" -code of the external libraries is cleaned up before distribution.[8]

- **How:** Bundling of the code is done with bundler applications, of which there are several available. For the JavaScript the most common ones are Browserify, Esbuild, Webpack, Parcel and Rollup.

- **Example:** For this example, we will use Webpack as our bundler. Webpack is a popular tool that can bundle, minify, and transpile your JavaScript code, among other things.

2.2.4 Data compression before transmission

- **Why:** From the viewpoint of green programming, all compressible data should always be compressed before it is transferred across the network, to reduce the strain and thus energy consumption. However, not all data is compressible, such as most images, videos, songs, any already compressed data, or encrypted data. It should also be noted that inline-compression may make the data transfer slower.

[8] Bundling

- **How:** Again, implementing the compression of data for transfer depends on the used framework and language, but usually it is in-built as an optional feature in to the code that facilitates the data transfer.
- **Example:** For this example, we will use Node.js with the Express framework to create a simple server that compresses the data before sending it. We will then use the compression middleware for Express, which provides gzip compression.

2.2.5 Choosing the right protocol and message format

- **Why:** Protocols and message formats used in data transfer differ in energy consumption, as some are more "talkative", in other words they use more headers and other extra information to facilitate the data transfer. From the viewpoint of green programming, protocols that achieve the same effect with less energy consumption should be used, if it is feasible. It should be understood that this is not always possible at all, although application specific -protocols have been proposed.[22]
- **How:** The exacts of how the protocol is changed to another depend on the used framework and language, but usually the protocol used in data transfer is either set in a configuration file, or it is passed as a parameter of the function or the object that facilitates the data transfer. See example for details.
- **Example:** In this example, we will use HTTP/2 usage. It is known that HTTP/1.1 opens a new connection for each request/response pair, which can be inefficient. HTTP/2, on the other hand, allows for multiple requests and responses to be multiplexed over a single connection, reducing the overhead and latency associated with establishing connections.

2.2.6 Eliminating presentation data transfer

- **Why:** In theory this means that only the contents of the application or website should be transferred from the database to the user with each interaction, and that the user interface -elements should be managed in the client -side.
- **How:** In practice this means using SPA (Single Page Application) -implementations, which are not possible with all frameworks or content management systems. Commonly used SPA -frameworks are for example ReactJS, AngularJS, VueJS and EmberJS, which are more or less similar JavaScript -based systems designed for smooth front-end development. The reason for this similarity is mainly due to fact that JavaScript is technically speaking the only viable way of creating SPA -frameworks, as of date.
- **Example:** In this example we will use React, a popular JavaScript library for building user interfaces, to demonstrate the usage of this practice.

2.2.7 Transmitting only changed data

- **Why:** Only the data that has been changed, i.e. data that has gained a different value than it had before, should be transmitted either way.
- **How:** In practice, the application must be aware of the changes happening and not submit data that has not been changed. For example, if you have a form which allows for the updating of the user information, and the user changes only one value, while the other values remain as they were when the form was loaded, then only that one variable which was changed should be transmitted back to the database. There are ready-made libraries for doing this, such as AG Grid's React Data Grid[9] for ReactJS.
- **Example:**For this example, we will create a simple form in Node.js and vanilla JavaScript, which tracks user input changes and only sends the modified data back to the server.

2.2.8 Identifying immutable data

- **Why:** Immutable data, in other words data that changes never or very rarely, should be kept in some kind of cached form so it would be available without transmitting it again every time the application or website is used. However, in this kind of scenario there must be a functionality to recognize when the data has to be reloaded.
- **How:** In abstract sense, this can be achieved by first conducting a review on the data that is fetched from the database, then identifying those data items that qualify as "immutable" in the sense we are talking about and then making a local copy of the identified immutable data item the first time it is fetched from the database. Then in the subsequent displays of the data this copy is called upon, instead of fetching the data again from the database. Example below demonstrates how this kind of feature is implemented in ReactJS.
- **Example:** In this example, we will create a JavaScript application using React for the frontend and Node.js with a MySQL database for the backend.

2.2.9 Checking data before transmission

- **Why:** Extensive checking of the user-inputted information, before it is transmitted over the network, is useful from the perspective of green programming as it reduces the amount of error messages the system generates.
- **How:** Input -elements can be controlled to filter incorrect input types quite easily. Usually this is done by directly adding certain attributes to the input -tag, but this can also be implemented with JavaScript. See the example for further information.

[9] AG Grid

- **Example:** In this example, we will use a simple HTML form with JavaScript for client-side validation.

2.2.10 Combining data for transmission

- **Why:** Simply for the reduction of network traffic, and thus energy consumption, it would be useful to bundle several messages in to one transmission, but only if the messages are reasonably combinable and if the end-product is easier to package than the messages would be by themselves.
- **How:** The code of the application must be factored so that suitable message transmissions are bundled in to one, if it is feasible. How this is exactly done is dependent on the used framework and language. See example for details:
- **Example:**In this example, we will use a web application where users can update their profile information and newsletter preferences. Instead of sending each piece of information in a separate request, we can combine them into a single request.

2.2.11 Minimizing HTTP headers

- **Why:** Large portion of communication by the modern applications happens over the HTTP -protocol, and it's quite common that the external libraries add custom HTTP-headers[10] to the requests, regardless of whether these are really needed or not. These should be reviewed and, if they are found to be useless for the purpose of the communication, removed to reduce the energy consumption.
- **How:** Google Chrome Developer Tools and similar browser -based observation applications can be used to view headers that are attached to every request the application makes. Other way to observe the headers, in the case of an application that is not accessed through the browser, would be to use the usual debugging features and print out the headers every time the application makes a call over the HTTP -protocol. There is also a package available through npm named `webhint`, which includes functionality to detect unneeded headers automatically.
- **Example:** In this example, we will create a simple Node.js server that serves a static file with minimal HTTP headers and manually analyse our server's response headers and ensure they are optimized.

2.2.12 Reducing HTTP redirection

- **Why:** HTTP redirection is in itself a useful technique that allows for redirecting the traffic to another address in situations where the original

[10] What is HTTP header?

address is unavailable, for example because of maintenance break. However, redirection is also quite resource-intensive operation, and while it's usually evaluated from the viewpoint of increasing the traffic, it should be understood that this increase in traffic always impacts the energy consumption. Thus, all HTTP -redirections happening in your application or website should be reviewed and if found unnecessary, deprecated.

- **How:** Exact details depend on the used language, and there are ready-made libraries to do this. For example, the npm package `webhint` includes a functionality to automatically detect and disallow HTTP -redirects.
- **Example:**In this example, we will show how to minimize redirection in a Node.js application.

2.2.13 Minimizing server-to-server data transfer

- **Why:** Almost always the server that hosts the application or website and the database which it uses are separate from each other. For the purposes of green programming, minimizing the traffic between these two is important, and can easily achieved by reviewing the database calls and reducing them to essentials.
- **How:** Database calls made by the resource must evaluated, and formulated so that only the necessary variables are fetched from the database. For example, if you need the name and the address of the user, you don't fetch the whole table row containing the user information, but only those variables which are needed.
- **Example:** In this example, We will create an optimized endpoint that fetches only specific columns from a user table instead of the entire row.

3 Green UI/UX Design

UI/UX -design is the aspect of the software development that deals with the design and implementation of user interfaces, and how easy and intuitive they are from the perspective of the end-users. In the past decade, we have seen the proliferation of this part of the industry, as ICT -technology has become the mainstay of the consumer electronics, which has required extensive streamlining and improvement in the ways in which applications can be used by people who for the most part don't know how the technology actually works, and could not use it without graphical interfaces and clickable icons. As the use of internet in everyday life has increased to be commonplace, the contents of the websites and applications are increasingly being presented both in clearer and more fancy ways. While this is in itself good, there are consequences to this development which should be kept in mind when designing things with the perspective of green programming in mind.

The other side of this coin is that with every improvement to the usability of software this has brought forth, it also increases the computational cost of these applications. More you have animated elements and slick effects that accompany each action, the more the computer uses processing power to conjure these electrical mirages to your screen. While some of this increase is justified in the name of bettering the user experience, quite often it's just eye candy that is meant to look nice, with no practical purpose whatsoever. On top of this, the UI/UX -design has spawned several outright malicious practices, such as the use of dark patterns, which are essentially ways to trick people to do something they wouldn't otherwise do, for example to give their consent to data collection. At worst, extensive use of dark patterns can lead to increased usage times by convolution of the controls the application offers, and thus to increased energy consumption.

In this chapter we will offer some perspectives on how to design user interfaces and experiences with environmental impact in mind, so that these techniques could be used better to achieve the aims of green software development. The techniques present here are just your average run-of-the-mill UI/UX design principles, but they are viewed through the design philosophy of how to use them in a way that results in less energy consuming outcome.

3.1 Difference between Green UI/UX Design and greenwashing

When researching this topic online on your own, you may come across articles and blog posts that talk about "green" or "sustainable" UI/UX design. Sometimes, they have good ideas, some of which have been used as inspiration for our materials also, such as this article. Sometimes, they have ideas that purport to be geared towards improving sustainability, while actually promoting practices that are anything but. For example, this article talks a lot about these ideas, but ultimately everything it teaches boils down to practices that will increase energy consumption, not decrease it. While many things mentioned there are impor-

tant, such as increasing inclusivity, diversity and eco-consciousness, the way to do it is not by creating "more ecologically minded animations". The way to do it is by decreasing the overall amount of animations, and other purely decorative designs. These things will not become more sustainable by them depicting "green ideas". This is a blatant example of "greenwashing", in other words of claiming to have sustainability in mind, but actually just creating diversions that are aimed at fooling people to think that something has been done to lessen the environmental impact.

3.2 Avoiding user errors

- **Why:** The less errors the users make in using the software, the less the software has to respond to these errors and thus less energy is consumed.
- **How:** Reducing the probability of human error in using a given software is not straightforward undertaking, and it requires a certain degree of design-in-advance. One way to achieve this is through minimizing the interaction the user has, in other words narrowing the scope of what the user can input and interact with, and thus decreasing the probability that the user does something that shouldn't have been done. Other way of approaching the issue is through the so-called co-design principle, where the end-users-to-be, i.e. representatives of the customer and other stakeholders participate in the design process of the application, as they often have a different perspective than the developers and designers in to how the application needs to be used.

3.3 Avoiding the use of dark patterns

- **Why:** Dark patterns are UI/UX -design choices which are meant to distract, mislead and outright fool the user of the application or website to make decisions which they wouldn't otherwise do, and which usually are somehow detrimental to them. Usually, but not only, dark patterns are used in the design of the cookie consent banners to trick the user to give consent to data collection facilitated with some kind of data analytical tool, such as Google Analytics.

  Abstaining from using the dark patterns in UI/UX -design is very important from the perspective of the green programming. Not only are dark patterns immoral and sometimes outright illegal, they contribute to increase the energy consumption of the application in several ways. First off, their deployment directly contributes to elevated use of website analytical tools, through misleading the users to give consent to data collection and thus activating the analytical tools used by the website. Secondly, they indirectly contribute to the energy consumption by adding elements that are ultimately unneeded and by complicating the user interface in attempt to hide functionalities or deter the user from doing something,

which needlessly increases the number of clicks and time spent with the website or application.

- **How:** Developers are strongly advised to study various sources documenting the most common dark patterns, such as the "Report of the work undertaken"[11] by the European Data Protection Boards' Cookie Banner Taskforce, which details some of the most common dark patterns. Developers should familiarize themselves with these design choices to better avoid them, as it has been observed that dark patterns do not emerge solely out of questionable intentions of the designers, but also occur naturally as a side-effect of human psychology.

3.4 Eliminating unnecessary elements

- **Why:** While designing green UI/UX, minimalism is in general a good guideline to follow. The less elements there are to render, the less energy is expended. Part of this striving for less elements is the reduction of purely aesthetic decorations which have recently become commonplace, such as the excessive use of on-mouse-hover effects, animations and especially background-effects such as video. It's true that our modern-day data transfer technology allows for the use of such cosmetic elements more than what was possible just a moment ago, but it should be kept in mind that such embellishments have a price paid in energy consumption as well.

- **How:** When designing and implementing the graphical outlook of the application or website, special consideration should be put in to what kinds of effects are essential for the improvement of the user experience, and which are not. Sometimes, for example, the on-mouse-hover -effects are really important for improved usability, to convey some kind of vital information for the user, but mostly they are deployed simply for no reason at all. Such pointless decorative use should be avoided.

3.5 Streamlining functionality

- **Why:** The shorter the path to the destination, less energy is expended in travelling to it. In the same way, less clicks and page renders it takes for the user to arrive at the content he seeks, the better, at least from the viewpoint of green programming. In practice this can take many forms, but often means the design of the UI/UX in such a way that excessive nested menus and link bars would not be needed.

- **How:** All applications and websites have features that fall in two categories; those which are used most of the time, and those which are used only rarely. Accordingly, those in the first category should be given priority treatment in such a way that accessing them is made easier. Common

[11] Cookie Banner Taskforce: Report of the work undertaken

way to do this is for example by using "lift-ups", through which the central functionalities are displayed in the landing page of the application, so that the user never needs to go searching for them from menus or link bars.

4 Measurements of power consumption

4.1 Measurement devices

4.1.1 AC -meters

The most common and easily available device for power consumption measurement is the AC -meter, which can be nowadays procured from any normal electronics appliance store. It is also very easy to use; the meter is just connected between the power supply and the device of which energy consumption is to be measured. As its' name implies, AC -meter can be used to measure the power consumption of any device that is powered with alternating current, which makes it very flexible for several different measuring projects.

While the scale at which the AC -meters operate is relatively wide, devices with very high power consumption may be beyond its' ability to measure. It's also relatively inaccurate compared to other measuring devices, which may present difficulties if used to measure software power consumption.

4.1.2 Bench power supplies

Bench power supplies are, as their name implies, power supplies used normally in laboratories and other special environments, which come with an integrated power measurement ability. Bench power supplies are extremely accurate but very limited in scope, being able to read only specific direct currents. Due to this specialized nature they are very expensive, and thus best reserved for situations where their high accuracy is required.

4.1.3 Meters connected to the DC -power supply

These types of devices are, together with AC- and USB -connected meters quite common and easily available. This kind of meter is connected to the direct current power supply, from which it channels the power to the device which consumption it is meant to observe, again quite similarly to how AC -meters function.

4.1.4 USB -connected meters

Like the AC- and DC -meters, this type of measuring device is also connected between the device of which energy consumption it is meant to measure and the power supply. Specifically this kind of meter is connected to the USB -power supply port of the device, and from it detects the amount of consumed energy.

4.1.5 Integrated power measuring circuits

Integrated power measuring circuits are literally electric circuits capable of measuring the power that passes through them. Their deployment and calibration is thus much more technical operation than with the other measuring devices mentioned here, and requires certain degree of expertise in the theoretical aspects

of power measurement. Pre-calibrated measuring circuits are also available, but this makes them only slightly easier to use than completely bare-bones circuits. In exchange for complexity in usage these components are relatively cheap, and can be integrated in to other devices to provide power measurement capability. As it's just a component, not a whole device, to process its' readings the power measuring circuit must be accompanied by either computer or micro-controller.

4.2 Measurement software

Using software to measure the energy consumption of the application is possible, but it's never as accurate as the best hardware solutions. It's however much easier to implement, especially for software-savvy developers who might lack electronics expertise, as setting up the energy measuring software is a trivial task for anyone with even a modest development experience, and because it does not require acquisition of any external resources - even the measuring applications are usually free. There are many different applications for measuring energy consumption, and the ones' presented here are just examples of this kind of technology.

The reasons for why there are no universal software based measuring tools for energy consumption boil down to the fact that to measure the energy consumption it's basically mandatory to have access to the hardware of the device - in other words, this requires the use of device or manufacturer specific interfaces such as Intel PCM (for Intel devices) which can communicate with the hardware components of the device. Thus, it's practically impossible to create one application that would function on all different types of devices. Theoretically, it could be possible to create a software that first detects the used device properties, and which would have interfaces defined for all of the possible devices, from which correct methods would be picked for use based on the device type, but thus far no-one seems to have taken up such a task. Some of the applications presented in this chapter are somewhat like this, for example they function on both Intel and AMD -processors.

Other aspect of the difficulty to create one-size-fits-all multitool for energy measurement comes down to differences in operating systems. While it's of course possible to create same application for different operating systems, and this is quite common these days, these are essentially slightly different versions of the same application from each other due to the differences inherent in the way operating systems and applications interact. It's technically possible to create an application that functions on all the common operating systems, but this is not very practical. In the case of energy measurement applications it might be quite impossible as they need to access process data that requires interaction with the operating system.

4.2.1 Intel PCM

Intel Performance Counter Monitor[12] is an API that allows the developer to create power-monitoring applications easily.[6] It's usable on Intel® Core™, Xeon®, Atom™ and Xeon Phi™ processors and on Linux, Windows, Mac OS X, FreeBSD, DragonFlyBSD and ChromeOS operating systems.

4.2.2 Syspower

Syspower[13] is a simple energy measurement tool for Mac. [6]

4.2.3 Website Carbon Calculator

Website Carbon Collector is a free-to-use, online tool which can be used for rough estimation of the website carbon footprint. It's very simple to use: you just input the wanted URL in to the calculator, and it gives you certain metrics of how much carbon is produced by the website in question, as well as some statistical information on how this compares to other websites globally. It also gives explanations of how the measurements are done. WCC API is free to use for non-commercial purposes, and can be provided free even for commercial uses, if the need to do so is reasonably argued for.

4.2.4 Windows Energy Estimation Engine (E3)

Windows Energy Estimation Engine is in-built tool in Windows operating system that allows for measuring the power consumption of applications.[14]

4.2.5 Powerstat

Powerstat is a Linux -only application developed by Colin King[15], based on Intel RAPL -interface, and is thus usable only on Intel -processors. It is used from the command line and provides the user with average estimations of power consumption in Watts, which must be taken in to account when measuring energy consumption in Joules. Either the user must run the calculations by hand or develop additional software which imports the results from the Powerstat and then calculates the energy consumption. [6]

4.2.6 PowerTOP

PowerTOP[16][17] is another open-source tool for measuring power consumption in Linux environment, usable both on AMD and Intel processors. PowerTOP is not only a power measuring tool, as it has functionalities to adjust the power

[12] Intel PCM
[13] Syspower git repository
[14] E3
[15] Powerstat
[16] PowerTOP
[17] PowerTOP git repository

consumption profile on the fly, and other advanced properties. Like the Powerstat it outputs the consumption in average Watts during a certain timeframe, from which the exact energy consumption must be further calculated. [6]

4.2.7 Perf

Perf[18] is another open-source tool for Linux/Intel -setup, based again on Intel RAPL. It is used from the command line and represents the output already computed to Joules', and it is capable of doing component -specific measurements, i.e. it can focus only on the RAM or CPU etc. [6]

4.2.8 Nvidia SMI

Nvidia SMI[19] is, unsurprisingly, a measurement tool developed by Nvidia corporation to be used with Nvidia GPU's. It is based on Nvidia Management Library, which similarly to Intel RAPL offers interface through which to monitor the workings and power consumption of the GPU. [6]

4.3 Measurement practices and procedures

4.3.1 Measuring the energy consumption with software

Regardless of which of the aforementioned applications you choose to use for measuring energy consumption, implementing the measurements to the development process is slightly more complex undertaking than just running the software and reading the results. Presented here is a method for running use-case scenario for two different versions of your software, and then comparing the results[6]:

- **Test scenario:** Prepare a reproducible use-case scenario in which to test the execution of your application. Ideally this is done by using automated unit tests.
- **Execution & measurement:** The scenario is executed while measuring the energy consumption on the background.
- **Identify problems in your code:** Based on the results the code is reviewed and improved in areas that seem to require further optimization. The points where the code requires optimization are not always obvious, and especially with further iterations this work becomes slower and slower, as the "excess to be trimmed away" becomes smaller and less visible.
- **Second test & measurement:** Test scenario is executed again, and the results are compared with the first execution. This cycle of executing the test scenarios and comparing their results against previous tests is continued until wanted threshold of energy consumption is reached, or until

[18] Perf
[19] Nvidia SMI

further optimization of the code does not affect the energy consumption anymore.

4.3.2 Measuring the energy consumption with hardware tools

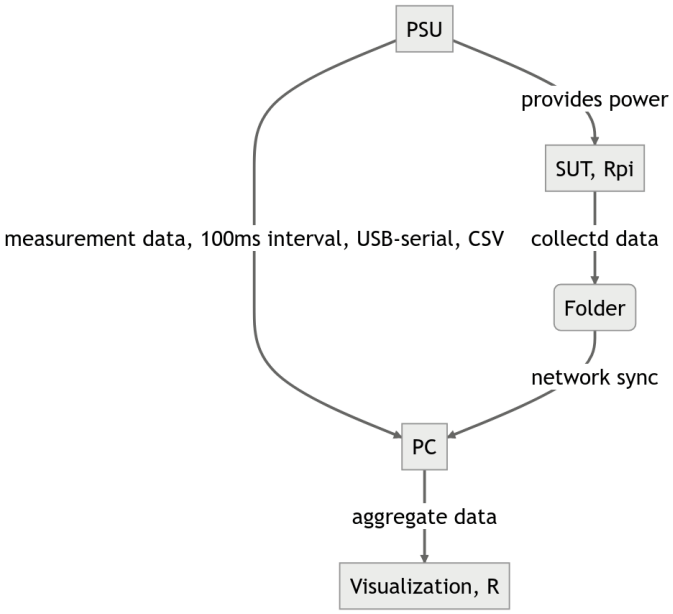

Figure 2: Simple measuring set-up for hardware meters

Presented in the Figure 2 is the simple measurement setup, which comprises of the following parts:

- **Power Supply Unit (PSU):** Power supply unit and attached meter. As can be seen from the left arm of the diagram, the meter is connected to the PC via USB -port.
- **System Under Testing(SUT):** The system that is being measured. It should be noted that in the architecture presented we are assuming that the measured software is run in external Raspberry PI (Raspberry PI(Rpi):). If you are setting up the measurements for a software sun on the PC, you can just ignore this side of the diagram entirely.
- **Folder:** A network drive which is connected to both the PC and the SUT.
- **PC:** The computer that collects the accumulated measurement data and coordinates the measurement device. It should be noted that the software whose energy consumption is being measured may also reside in the PC.

- **Visualization, R:** The visualization of the measurement data, implemented in R language.

From this architectural layout can be seen the basic principles of using the physical measurement devices of the type that is connected to the power supply unit (AC, DC and USB -meters). Basically what is needed is only the measuring device, the computer that runs the software which is to be measured, and scripts for handling the measurement data and visualizing it in to human-readable form, which are trivial to develop. The process of using this setup to assess the energy consumption is essentially the same as the one described in the section 4.3.1.

5 Code Optimization

When talking about code optimization, it's central to understand that only small part of the actual optimization is what we traditionally think of when we form a mental image about optimization. It could be said that this "real optimization", i.e. fine-tuning the algorithms to actually be better performing, and thus less energy-consuming, is less than 10% of the actual optimization. The rest of the optimization could be called "non-pessimization"[20]. That is, the principle of not writing code that is obviously heavier to perform than it should be. This might sound like stating the obvious, but in practice and reality, large portion of all code that is written could have been done better in the first place. While there are best practices and other guidelines on how something should be done, in the end of the day timetables, "good enough" -mentality and the effectiveness of the hardware that forgives excess resource usage means that as long as something is reliable and works at least relatively well performance-wise, it is good enough for production. This kind of "pessimization" process means that the benefits of the real optimization done on top of such development are, at best, very thin and that is why the "non-pessimization" of the development process is the first step towards actual optimization. It's also very important to understand that there is lots of "fake optimization" happening all the time, mostly because developers have erroneous ideas about what is useful and what is wasteful, what affects the performance and what does not, but also because the nature of ICT -business makes people reluctant to admit that they would not understand some technical aspect.

DISCLAIMER: Things that will be discussed later in this chapter, such as loop parallelism and interchange are things which most modern compilers perform automatically to optimize the bad code written by the programmer. That, however, doesn't mean that they couldn't or shouldn't be done by the programmer also, and for this reason we will present the basics of these techniques for your edification.

IMPORTANT: Always remember that to really know whether the optimization you've done actually improves the energy-efficiency of the application the results must be measured, techniques for which are provided in the chapter 4.

5.1 Non-pessimization

The very point of "non-pessimization" is to not write bad code. Simple, isn't it? In practice, this is not as simple as it sounds, but it's not rocket science either. The main theme throughout this course has been the fact that with the improvement of the computing power, memory and data transmission, code size has bloated as the modern hardware is so forgiving of writing a code that is far from optimized. That is, it's very pessimized. At the heart of the "non-

[20] Video: Philosophies of Optimization

pessimization", then, would be the trimming away of the completely needless excess code, much in the way that has already been partially discussed in the Section 2.2.1 and the avoidance of doing pointless things in the code.

Lots of this kind of optimization has already been discussed earlier in the course material, especially in the chapter 2. There we provided several examples of what kind of very basic optimization procedures should always be done to cut down the excess energy consumption. In this chapter, we will focus more on the mentality and attitudes that breeds those situations where the bad code comes to be, and what should be done differently to avoid this.

There are several common excuses for not paying too much attention to the performance quality of the code, which probably everyone who has programmed for even a small amount of time has heard[21]:

1. **No need** - Because modern hardware can accommodate atrociously bad code, optimization is eschewed.

2. **Too small impact** - The benefits of optimization are seen as too small versus the effort needed.

3. **Not worth the effort** - Basically the same rationalization as above, but more focused on the economical bottom line.

4. **Niche** - Optimization is seen to be useful only in very small fraction of situations or in some specific aspect of software development, such as gaming industry.

5. **Hotspot** - Optimization is seen to be required in special "hotspots" of the code, and thus it's mentally relegated to the "optimization specialists" so that the regular developer wouldn't have to think about it.

As we can see here, the first point in the list is already familiar from the previous chapters of this course material. The fact that modern computers can accommodate totally unoptimized code and everything is still running relatively fine, at least usually, is de facto biggest reason for the current situation. The second and third points are a bit more specific, but also related to the issue of the hardware situation as without very effective machines, no-one would think like this. Without the capacity to run very unoptimal implementations, lack of optimization would be directly felt in the bottom-line, and would be encouraged and even enforced by the managerial layer.

Ultimately they all boil down to dodging the issue of optimization by invoking reasons which, in the light of the optimization efforts undertaken by some of the largest software companies in the last few years, can only be seen as patently false.

[21] Video: Performance Excuses Debunked

5.2 Algorithm Design & Analysis

Design of better algorithms is one of the key factors in achieving better performance, and energy-efficiency, of the application. However, study of algorithms is a whole discipline of it's own, and one that we can't explain in depth in the scope of this course. Thus we will present the very basics of algorithm design and analysis here, and encourage the aspiring student to seek further education in these matters from the basic algorithm design course available here.

5.2.1 Asymptotic Complexity

Central to the understanding of the properties of the algorithm is asymptotic complexity, which is used to express the time and space limitations(or complexities) of the algorithm. It does not depict the real-world scenario of how the algorithm behaves on a specific machine X, but it gives a very useful idealistic representation of the fundamental properties of the algorithm. In this way, it's possible to see the relation between the running time and the input size of the algorithm, in the case of time complexity, and thus compare different algorithms with each other to determine which of them performs best. Space complexity, while being measured with the essentially same notation as the time complexity, is used to express the relationship between the input size and the memory required, but its' calculation will be left out of this exposition for now.

Big O

Ordo notation, commonly known as big O, is the mathematical way to present the asymptotic upper bound of the algorithm. There are several other asymptotic notations as well, but the big Ordo is the most commonly used, and thus most important to understand.

Describing the Ordo notation in detail would involve some amount of mathematics, which we will omit here, and go straight to simplest possible explanation of how it can be determined from the computer code:

- **Break the algorithm into individual operations**

  For example, this function has four individual operations:

```
function subtract(a, b) {
    let c = a - b;
    return c;
};
```

  - looking up the value of a
  - looking up the value of b
  - calculating a - b and assigning it to c
  - returning c

- **Calculate the O of each operation**

  In the example above, O of each operation is 1, simply because all of them are performed only once, and is expressed as O(1). This sounds deceptively simple, because in essence it is, as when the operations become more complex, it affects their O value. For example, a simple FOR loop is O(n), as it is performed n number of times. Likewise, two nested loops have the O value of n^2, or O(n^2).

- **Add up the O of each operation together**

  In our example, this yields O(4).

- **Remove the constants & find the highest order term**

  By removing the constants is meant that the result is reduced to it's highest order terms. In this case it means that O(4) becomes O(1). For the sake of time complexity, the constant values of the function don't matter - O(13) and O(300) both boil down to O(1).

- **Result**

  In this example, our result is O(1), which means that the algorithm performs in constant time.

Common Time Complexities

Following is a list of different time complexities from the simplest to the most complex with typical examples.

- **Constant** O(1)
  Example: See the example used in this chapter. In general, all basic mathematical operations of 64-bit integers perform in constant time, unless the used numbers are astronomically high.

- **Logarithmic** O($logn$)
  Example: Binary Search[22]

[22] Binary Search

- **Linear** O(n)
  Example: FOR loop
- **Loglinear** O($nlogn$)
  Example: Quicksort[23], Heapsort[24], Merge Sort[25]
- **Squared** O(n^2)
  Example: Nested FOR loop
- **Cubic (and larger)** O(n^3), O(n^4), O(n^5) etc...
  Example: Thrice (or more) -nested FOR loop
- **Exponential** O(n^n), O(2^n) etc...
  Example: Recursive implementation of the Tower of Hanoi problem[26]

In most of the cases (there are exceptions), if your algorithm is larger than cubic in complexity, you're doing something very wrong. As a rule of the thumb, closer you get to loglinear, or even smaller complexity, the better.

5.2.2 Comparing the algorithms

Comparing two different algorithms, in other words determining the value of input n where other of them starts to exceed the other in time complexity, is relatively simple:

1. **Calculate the O values of the function:**
   For example, this function

```
function example_one(input a, input b) {
   int x = 0;
   int y = 0;

   for(int i = 0 ; i < a; i++) {
       x = x + a + b;
    }
   for(int j = 0; j < b; j++) {
       y = y - a - b;
    }

   print(x)
   print(y)
};
```

[23] Quicksort
[24] Heapsort
[25] Merge Sort
[26] Tower of Hanoi

would be 2n + 6, because: 2 x loops of input length n with one operation each = 2n and 6 basic operations = 6

This function, on the other hand:

```
function example_two(input c) {
    int z = 0;
    for(int i = 0 ; i < c; i++) {
        for(int j = 0; j < c; j++) {
            z  = z + i * j;
        }
    }
    print(z);
};
```

is $n^2 + 3$, because: one nested loop of one operation = n^2 and 3 basic operations = 3

2. **Comparing the functions:**
From this alone we can already see that the 2n + 6 is of **linear time complexity**, as it's highest order term is n, while $n^2 + 3$ is of **quadratic time complexity**, as it's highest order term is n^2. Thus, we can know that $n^2 + 3$ is always in principle more complex than the 2n + 6. However, with really small values of n, it might be that $n^2 + 3$ has a slower growth rate than 2n + 6. To know these values we form a expression and solve it:

$2n + 6 = n^2 + 3$

$-n^2 + 2n + 3 = 0$

$n = \frac{-2 \pm \sqrt{2^2 - (4 \cdot -1 \cdot 3)}}{-2}$

$n = \frac{-2 \pm \sqrt{16}}{-2}$

$n = -1$ or $n = 3$

Because n can't have negative value in real world (input can't be negative), we discard the n = -1. From this we can see that if the input is equal or smaller than 3, the $n^2 + 3$ indeed has a slower growth rate. In practice this is irrelevant.

5.3 Dependencies in code

Before we go in to the exciting landscape of optimizing loops, we have to take a brief peek at the dependencies that exist in code and loops, for understanding these things has certain importance for the practical application of the techniques that will be discussed later.

Type	Notation	Description
True (Flow) Dependence	`S1 ->T S2`	A true dependence between S1 and S2 means that S1 writes to a location later read from by S2
Anti Dependence	`S1 ->A S2`	An anti-dependence between S1 and S2 means that S1 reads from a location later written to by S2.
Output Dependence	`S1 ->O S2`	An output dependence between S1 and S2 means that S1 and S2 write to the same location.
Input Dependence	`S1 ->I S2`	An input dependence between S1 and S2 means that S1 and S2 read from the same location.

Figure 3: Forms of code dependencies

As we can see from the above figure[27][32, 11], there exists four types of dependencies in code: True(Flow), Anti, Output and Input Dependencies. Below are presented examples of these different dependencies:

- **True Dependency**

```
int x, y;
x = 1;
y = x + 100;
```

  If an operation writes to a variable that is later read by another, they have True Dependency.

- **Anti Dependency**

```
int x, y = 30;
x = y - 50;
y = -2;
```

  If an operation reads from a variable later written to by another operation, they have Anti Dependency.

- **Output Dependency**

[27] Loop-level parallelism

```
int a, b = 40;
a = b - 38;
a = 2;
```

If two operations write to same variable, they have the Output Dependency.

- **Input Dependency**

```
int a, b, c = 2;
a = c - 1;
b = c + 1;
```

If two operations read from same variable, they have Input Dependency.

5.3.1 Dependencies of loops

Dependencies in loops are of two forms, **loop-carried dependence** and **loop-independent dependence**. The difference between the two is that in the loop-independent dependence, each iteration has dependence only with the variables within the iteration, i.e. it has inter-iteration dependence. This kind of loop can be run in parallel without worrying about synchronization.

In the loop-carried dependence, on the other hand, there is some kind of dependence between the statements that exist in different iterations of the loop. In other words, the statements of the loop are dependent of the output of the statements of the previous iteration of the same loop.

5.4 Loop interchange

When working with nested loops, it should be understood that the loop order can often be changed without affecting the correctness of the algorithm. This process is known as loop interchange which, while it might feel a little counter-intuitive, may actually have a very significant effect on the performance, and thus the energy consumption, of the application. The mechanism of what happens here, and how is this performance enhancement possible in the first place would require explaining the compiler theory at length, but it suffices to say that the reason lies in the order in which data is stored in the memory, and that using loop interchange avoids executing cache misses which are costly performance-wise.

For example, the following:

```
for (x = 0; x <= 100; x++)
  for (y = 0; y <= 50; y++)
    a[x,y] = x * y
```

can be written as:

```
for (y = 0; y <= 50; y++)
  for (x = 0; x <= 100; x++)
    a[x,y] = x * y
```

Important thing to remember when considering the loop interchange is that while doing so, the loop dependencies must be kept intact. That is to say, when interchanging the inner loop with the outer (or the middle loop layers in the case of multiple nested loops), care must be taken to account for dependencies in the code, and especially in the loop structure, so as to not cause any bugs. It should also be noted that while the loop interchange can improve the performance, it can also decrease it if done incorrectly. In practice the results must always be measured after performing the loop interchange to know if the energy efficiency actually was increased.

5.5 Parallel loops

Applying multi-threading to loop execution can significantly decrease the execution time and potentially increase the energy-efficiency of the application. In the case of simple loops, implementing this parallel processing can be embarrassingly simple, as it's trivial to assign separate thread for every iteration, but with more complex algorithms there must be taken extra steps to ensure that sequential nature is not broken. There are many different ways of applying the loop parallelism in practice, of which two will be explained with examples below. Other techniques also exist, but due to them being several degrees more complex and this course being from beginner-to-intermediate degree in difficulty, we have chosen these two to illustrate this kind of optimization.

5.5.1 Loop parallelism methodologies

- **DISTRIBUTED Loop**

  Let's consider the following:

  ```
  for (int i = 1; i < n; ++i) {
      x[i] = x[i-1] + y[i];
      z[i] += r[i];
  }
  ```

  Since the operations don't have loop-independent dependency, meaning that they are not dependent on each other, we can split the original loop as such:

  ```
  for (int i = 1; i < n; ++i) {
      x[i] = x[i-1] + y[i];
  ```

```
}
for (int i = 1; i < n; ++i) {
    z[i] += r[i];
}
```

Then, these two loops can be run in parallel, a practical example of which is provided below.

Example: In this example, a simple java code example will demonstrate the usage of DISTRIBUTED loop.

- **DOALL Parallelism**

  This form of parallelism is possible in situations where the statements within the loop don't have a loop-carried dependency, as in they are not dependent on the earlier iterations of the loop. In this case all iterations can be assigned to their own threads, and run in parallel. Consider the following code:

  ```
  for (int i = 0; i < n; ++i) {
      x[i] = y[i] + z[i];
  }
  ```

  Array `x` is not read by any iteration, and arrays `y` and `z` are not written to by the iterations. Thus no iteration of the loop has dependency on any other iteration of the loop. In this kind of situation it's possible to implement DOALL Parallelism, in which all of the iterations are assigned their own thread and run in parallel.

  Example: In this example, a simple java code example will demonstrate the usage of DOALL Parallelism.

5.6 Loops or list methods?

Many programming languages offer ready-made tools to iterate over the contents of arrays and similar structures, which essentially can be used to do what is also done with loops. The question arises, whether these methods are better, performance-wise, to just using traditional loops? As with all things in this course, the answer is not exactly clear or conclusive. The end result varies heavily based on the language that is used and even the specific method in some cases.

For example, in Python, the list comprehension and filter function are much more efficient than for loops.[28] On the other hand, in the case of JavaScript, it seems that loops perform slightly better in this regard.[29] Reasons for this boil down to how well these methods have been optimized in the first place.

[28] For Loop vs. List Comprehension
[29] Array method or a loop?

5.7 Using AI in code optimization

In recent years, LLM[30] -type AI's have become publicly available, and due to their properties have been adopted widely for various tasks. One area where GPT-4 and similar technologies are quite good at is writing blocks of code, and also in optimizing existing code. This has led to development of specific software development tools based on this technology, such as Github Copilot[31].

From the perspective of green software development, such tools are a two-edged sword. They do allow for hastening the development work significantly, and they also allow for, especially for junior developers, the optimization of code with ease. At the same time, these kind of systems use relatively high amount of electricity, which increases the carbon footprint significantly. Thus it's not exactly clear whether using them for code optimization, even when this optimized code in itself would be less energy-consuming, is ultimately beneficial. On one hand, we could infer that if the developed software to be optimized by these tools is "big enough", in other words it will be used for long time and by enough people, this offsets the carbon footprint of the usage of AI in the long run. On the other hand, if such tools are used for every small project by millions of developers around the world, the end result may be negative in terms of environmental impact. As this development is quite recent there is thus far very little, if any, research done on to this situation, so we must work with assumptions based on our previous knowledge of the energy consumption of these systems and software development in general.

6 Green Cloud Services

In the past two decades, cloud computing has steadily grown in prominence and has by now become one of the main aspects of the modern ICT -industry. From the perspective of green programming this is a positive thing; cloud computing, in general, is much more resource- and energy -efficient way of organizing the computational infrastructure and resources than the preceding paradigm of personal computers and localized infrastructure, as was shown in the study by Park et al. [25]. However, in itself the "traditional" cloud computing is still very energy-intensive practice, and while the environmental impact as a whole might have lessened somewhat by the widespread adoption of this paradigm, data centers used in the cloud computing still account for roughly 1% of the global annual carbon footprint.[32]

To make the cloud computing more green, several different techniques have been devised. Foremost among these is the construction of the sustainable data centers, which are built from the beginning to be as energy-efficient as possible. Such facilities are constructed with low-emission building materials, employ renewable energy sources as much as is possible, are designed so that

[30] Large Languate Model
[31] Github Copilot
[32] Data Center Energy Consumption

the energy consumption in their operation and maintenance is minimized and where special consideration is put into the thermal management, as the cooling down of the servers is one of the largest contributors to the carbon footprint of these kinds of installations. [30] However, this kind of approach to reducing the carbon footprint is possible only through the construction of new data centers, or through slow and costly upgrading of older data centers. and for this reason other techniques of lessening the environmental impact are also needed.

Apart from and beyond the construction of new data centers following the green principles, there are many other ways of improving the energy-efficiency of cloud services. Server power consumption can be minimized with better management, algorithms can be deployed with the aim of managing the use of virtual machines more efficiently and balancing the workload between servers more evenly, and even older centers can switch to using renewable energy.

6.1 How to compare services?

From the viewpoint of the developer, or the leader of the development team, the previous section doesn't answer the most important question, namely "How to choose the greenest option?". How to even compare different service providers to determine which one of them is the "most green"? Truth be told, there isn't exactly any scientific consensus on the topic, but according to the Jonathan Koomey[33], whom can be considered something of a scientific expert on this issue, quoted by the Wired[34], there are three factors that should be taken in to account:

- Efficiency of a data center's infrastructure (lights, cooling, and so on)
- Efficiency of its servers
- Source of its electricity

Google, Microsoft Azure and Amazon Web Services (AWS) are the veritable titans of cloud computing, together accounting for two thirds of all cloud services, with AWS alone being larger than its two largest competitors together. Of these three, Google is greenest by far and AWS the least. Not only has Google invested three times more than Amazon on renewable energy production, but it also is several levels more transparent about its energy consumption and the means of decreasing it. Microsoft Azure follows close on the heels of Google, losing the race for greenest cloud provider only because its portfolio of renewable energy is considerably smaller. Then again, Google is the "smallest" of these companies when it comes to production output of the cloud services.

It should also be noted that while all of these corporations claim or at least attain to achieve carbon neutrality, none of them actually achieves this goal as of yet, without the buying of renewable energy certificates (REC), which are

[33] Jonathan Koomey
[34] Amazon, Google, Microsoft: Here's Who Has the Greenest Cloud

criticized for distorting the understanding of how environmentally friendly different actors are.[2] With the RECs factored in to the emissions, both Microsoft and Google are 100% carbon neutral, and Amazon 50%, but without accounting the RECs these numbers fall considerably. There is no clear data on how much this affects the carbon neutrality of Google, but it's known that Microsoft data centers are only 60% carbon neutral without RECs, and AWS considerably less than that.

Tech giants have approached the goal of going green with their cloud services in slightly different ways. Google has developed extensive AI -powered monitoring systems that constantly survey the weather conditions and other environmental variables around the data centers and dynamically adjust thermal controls based on this information. Microsoft, on the other hand, has a quite unique internal carbon tax system, with which they encourage their departments to find greener solutions using economic incentives. While all of these corporations have adopted the practice of building data centers to colder climates, Microsoft has also started to experiment with building data centers underwater, to better utilize the environmental cooling factor. AWS is the most secretive of the three, and is known for not answering questions about how they utilize newest innovation to solve the dilemma of reducing the carbon footprint.

6.2 Tools to measure the carbon footprint of the cloud service

All of the large cloud providers have released several different tools for their customers to aid in real-time monitoring of the carbon footprint of the used services. In addition to this, several third-party carbon footprint -tracking tools for cloud services exist, such as the free open source solution Cloud Carbon Footprint[35].

AWS has the Customer Carbon Footprint Tool[36], which allows the users of the AWS cloud services to track and forecast their carbon footprints. Google has a similar tool called Carbon Footprint[37], and it also offers another tool which allows the users to compare the carbon footprints of available cloud services, and pick the one which has the least enviromental impact[38]. Google also has an AI-powered Unattended Project Recommender, which automatically recognizes unused and abandoned code in the servers the customer is using, and prompts for its' removal. Microsoft offers several different tools for their customers towards these ends, one being the Emissions Impact Dashboard[39], which is similar to the tool AWS has.

Third-party systems that are meant for measuring the environmental impact of the cloud computing are either free and open source or developed by smaller ICT -companies which themselves usually don't offer cloud services. The afore-

[35] Cloud Carbon Footprint
[36] AWS: Customer Carbon Footprint Tool
[37] Google: Carbon Footprint
[38] Helping you pick the greenest region for your Google Cloud resources
[39] Emissions Impact Dashboard

mentioned open source Cloud Carbon Footprint, for example, is compatible with all of the three large operators, and provides the user with various graphical and numeric ways to observe the carbon footprint.

6.3 Green cloud optimization

Extensive use of virtual machines and the consolidation of physical servers it allows is one of the key strategies of achieving energy-efficient cloud computing. On the other hand, it's often claimed that distributing the computation across multiple physical units, i.e. implementing load-balancing algorithms, decreases the overall energy consumption, but the truth is not so straightforward. Depending on the overhead, this may actually increase the energy consumption. However, load-balancing does increase the lifespan of the physical servers as it reduces the strain they endure, and in such a way contributes towards the green cloud optimization.

In principle there are two kinds of applications running in the cloud services; those that are constantly active, and those that are activated only when they are called by the client. If your application is of the first type, it should be factored so that all the operations which should anyway be done at some point of the run are done immediately after the launch. This means initiating the database connections, setting the configurations etc. If it's of the second type, only the absolutely necessary operations should be performed, and everything else be initiated only when it's necessary to do so.

Data deduplication[40], i.e. the elimination of the multiple copies of the data, and systematic compression of all stored data should be practiced to make the cloud greener. Other way to optimize the energy consumption of the data storage is to use tiered storage, in such a way that frequently accessed data is stored in high-performance drives, while rarely used data is located in storage-tiers that consumes less energy.

6.4 Avoiding Cloud Overflow

One aspect of cloud usage that ties to both the optimization of the cloud itself, but also the broader topic of optimization in general, is avoiding the "cloud overflow"[41]. And by avoiding, we mean avoiding it at all costs. In a nutshell, badly optimized algorithms and bad algorithm design in general, combined with careless configuration of the cloud platform, can in the cloud environment lead to a situation where the cloud infrastructure spawns endless amount of virtual servers to accommodate for the increase in computational demand. When the software is running on one computer, careless use of recursion can lead to stack overflow. When the same recursion runs in the cloud, only the sky is limit to the problems this can cause. While the effect on the user of the cloud is mainly that the expenses will go skyrocketing in extremely short time-frame, it will also increase the environmental impact of the cloud usage completely needlessly.

[40] Data Deduplication
[41] Video: Cloud Overflow

Many cloud providers don't provide automatic kill switches to prevent this kind of disaster from happening, either.

References

[1] Luca Ardito, Giuseppe Procaccianti, Marco Torchiano, and Antonio Vetrò. Understanding green software development: A conceptual framework. *IT Professional*, 17(1):44–50, 2015.

[2] Anders Bjørn, Shannon M Lloyd, Matthew Brander, and H Damon Matthews. Renewable energy certificates threaten the integrity of corporate science-based targets. *Nature Climate Change*, 12(6):539–546, 2022.

[3] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. Is software "green"? application development environments and energy efficiency in open source applications. *Information and Software Technology*, 54(1):60–71, 2012.

[4] Shaiful Alam Chowdhury and Abram Hindle. Greenoracle: Estimating software energy consumption with energy measurement corpora. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, page 49–60, New York, NY, USA, 2016. Association for Computing Machinery.

[5] Marco Couto, Rui Pereira, Francisco Ribeiro, Rui Rua, and João Saraiva. Towards a green ranking for programming languages. In *Proceedings of the 21st Brazilian Symposium on Programming Languages*, SBLP '17, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Luís Cruz. Tools to measure software energy consumption from your computer. http://luiscruz.github.io/2021/07/20/measuring-energy.html, 2021. Blog post.

[7] Waltenegus Dargie. A stochastic model for estimating the power consumption of a processor. *IEEE Transactions on Computers*, 64(5):1311–1322, 2015.

[8] João De Macedo, Rui Abreu, Rui Pereira, and João Saraiva. Webassembly versus javascript: Energy and runtime performance. In *2022 International Conference on ICT for Sustainability (ICT4S)*, pages 24–34, 2022.

[9] Markus Dick and Stefan Naumann. Enhancing software engineering processes towards sustainable software product design. In *EnviroInfo*, pages 706–715. Citeseer, 2010.

[10] Giorgos Fagas, John P Gallagher, Luca Gammaitoni, and Douglas J Paul. Energy challenges for ict. *ICT—Energy Concepts for Energy Efficiency and Sustainability*, pages 1–36, 2017.

[11] Gina Goff, Ken Kennedy, and Chau-Wen Tseng. Practical dependence testing. *ACM SIGPLAN Notices*, 26(6):15–29, 1991.

[12] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867, 2021.

[13] Michael Z Hauschild, Ralph K Rosenbaum, and Stig Irving Olsen. *Life cycle assessment*. Springer, 2018.

[14] Abram Hindle. Green software engineering: The curse of methodology. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 5, pages 46–55, 2016.

[15] Takuro Inoue, Makoto Ikeda, Tomoya Enokido, Ailixier Aikebaier, and Makoto Takizawa. A power consumption model for storage-based applications. In *2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 612–617, 2011.

[16] Eva Kern, Markus Dick, Stefan Naumann, and Tim Hiller. Impacts of software and its engineering on the carbon footprint of ict. *Environmental Impact Assessment Review*, 52:53–61, 2015.

[17] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K Nurminen, and Zhonghong Ou. Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 3(2):1–26, 2018.

[18] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green algorithms: Quantifying the carbon footprint of computation. *Advanced Science*, 8(12):2100707, 2021.

[19] Simon Kemp Laurence A Wright and Ian Williams. 'carbon footprinting': towards a universally accepted definition. *Carbon Management*, 2(1):61–72, 2011.

[20] Adam Wade Lewis, Soumik Ghosh, and Nian-Feng Tzeng. Run-time energy consumption estimation based on workload in server systems. *HotPower*, 8:17–21, 2008.

[21] DA Maevsky, EJ Maevskaya, and ED Stetsuyk. Evaluating the ram energy consumption at the stage of software development. *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, pages 101–121, 2017.

[22] Rex Min and Anantha Chandrakasan. Mobicom poster: top five myths about the energy consumption of wireless communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1):65–67, 2003.

[23] Adel Noureddine and Olivier Le Goaër. Software energy efficiency: New tools for developers. *Ethical Software Engineering*, page 17, 2022.

[24] Candy Pang, Abram Hindle, Bram Adams, and Ahmed E. Hassan. What do programmers know about software energy consumption? *IEEE Software*, 33(3):83–89, 2016.

[25] Jiyong Park, Kunsoo Han, and Byungtae Lee. Green cloud? an empirical analysis of cloud computing and energy efficiency. *Management Science*, 69(3):1639–1664, 2023.

[26] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, pages 1–6, 2011.

[27] Giuseppe Procaccianti, Antonio Vetro', Luca Ardito, and Maurizio Morisio. Profiling power consumption on desktop computer systems. In *Information and Communication on Technology for the Fight against Global Warming: First International Conference, ICT-GLOW 2011, Toulouse, France, August 30-31, 2011. Proceedings 1*, pages 110–123. Springer, 2011.

[28] Greenhouse Gas Protocol. Greenhouse gas protocol. *Sector Toolsets for Iron and Steel-Guidance Document*, 2011.

[29] Reinder Radersma. Green coding: Reduce your carbon footprint. *Ethical Software Engineering*, page 19, 2022.

[30] Laura-Diana Radu. Green cloud computing: A literature survey. *Symmetry*, 9(12), 2017.

[31] John Reap, Felipe Roman, Scott Duncan, and Bert Bras. A survey of unresolved problems in life cycle assessment: Part 2: impact assessment and interpretation. *The International Journal of Life Cycle Assessment*, 13:374–388, 2008.

[32] Yan Solihin. *Fundamentals of parallel multicore architecture*. CRC Press, 2015.

[33] R. van Diemen, V. J.B.R. Matthews, J.S. Möller, V. Fuglestvedt, C. Masson-Delmotte, A. Méndez, S. Reisinger, and Semenov (eds). Ipcc, 2022: Annex i: Glossary.

[34] Max Van Hasselt, Kevin Huijzendveld, Nienke Noort, Sasja De Ruijter, Tanjina Islam, and Ivano Malavolta. Comparing the energy efficiency of webassembly and javascript in web applications on android mobile devices. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, pages 140–149, 2022.

[35] Antonio Vetro, Luca Ardito, Maurizio Morisio, Giuseppe Procaccianti, et al. Monitoring it power consumption in a research center: seven facts. In *Proceedings of ENERGY 2011, The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 64–69, 2011.

[36] Thomas Vogelsang. Understanding the energy consumption of dynamic random access memories. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 363–374, 2010.