Fuller Fellowship 2021

Green Software Engineering — Developer Guide

Contents

Introduction

Green Software Engineering aligns with my interest in software engineering, while also aligning with my (and Fuller's) interest in being more environmentally responsible, and more active when it comes to curbing our digital carbon footprint.

The goal of this publication is to inform the reader (specifically the development team) of the eight principles of Green Software Engineering as outlined by Asim Hussain, Green Cloud Advocacy Lead at Microsoft.

Iit is my hope that these best practices will place our development team, and Fuller, in a more environmentally responsible position when it comes to software development and utilising external cloud based resources.

Marko Rapaic

The Problem

As web developers, thinking about the environmental impact of every resource we use while developing websites doesn't necessarily come naturally to us. It's very easy to ignore the carbon impact of the choices we make in favour of making choices that save us time and money.

Garnier writes that the reality is that everything has a cost — our computers, the data centres we download resources from, the servers we host our websites on — which means everything has a carbon impact.

> "Due to software's ethereal nature, it's hard to visualise how it impacts our environment."

According to Garnier, the Information and Communications Technology (ICT) sector is growing at an astounding rate, consuming more than two percent of global emissions — practically the same as the aviation industry's carbon footprint from fuel emissions.

As a result, the impact has become too large to disregard, and the impact will only worsen with the development and distribution of more devices, data centres and other related technologies.

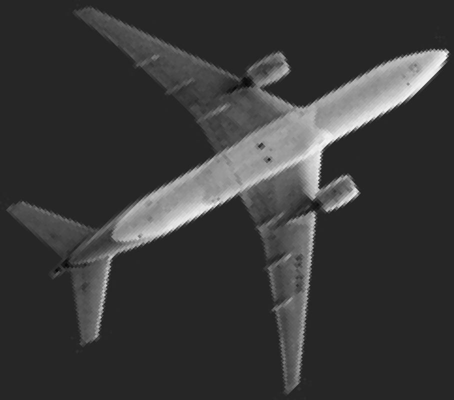
However, it's not too late. With the right mix of research and planning, there are ways we can initiate change for good.

More Devices Require More Energy

As stated by Hawkins, although the ICT sector currently makes up two per cent of global emissions, if the growth in digital consumption continues on the same trajectory, by 2040 the emissions will make up 15 percent of global emissions. This is the same as half of the world's transportation sector emissions.

Part of the reason for this explosive growth is the Internet of Things (IoT) – connected physical objects exchanging data over a wireless network. Juniper Research reported that the number of IoT connections will rise from 35 billion in 2020 to 83 billion in 2024, which is a 130 percent growth in just four years. And by 2023, 66 percent of the global population will have internet access (compared to 51 percent in 2018).

As this trend continues, a very real consideration is whether or not there will actually be enough electricity to power this seemingly endless increase of devices (not to mention data centres). Current sources of electricity (fossil fuels and nuclear power) are not ideal and while renewable energy sources such as wind and solar are growing rapidly, they may not be in a position to pick up the slack in the next decade.



Data Centre Proliferation

The rise in the number of devices and cloud-based services has led to an expansion of data centres, which consume two percent of the world's electricity. By 2030, that number could be eight percent.

Hawkins says "Data Centre Alley" is a little-known tech hub that processes 70 per cent of global internet traffic. Data Cenre Alley is home to a cluster of data centres, businesses, and government organisations in Ashburn, Virginia.

Companies like Amazon, Google, Apple, and Microsoft own or rent data centres in Data Centre Alley. Amazon dominates the world of cloud-computing real estate, owning enough data centres in the area that nearly a third of the internet runs on Amazon Web Services (AWS). Because of this, many tech companies and their stakeholders want their cloud-computing services to be hosted close by, for the quickest service and updates.

However, Hawkins says that what tech companies don't realise is while AWS may try to dedicate resources to powering their operations with renewable energy, the source of electricity for data centres is completely dependent on the local region. In this case, Data Centre Alley is reliant on Loudoun County, Virginia's resources: a utility provider called Dominion Energy that happens to run on mostly fossil fuels.

Many tech companies are then faced with a conundrum: choosing between high performance service and sustainability.

As of 2020, AWS has established five carbon-neutral zones where they purchase carbon offsets to balance the emissions coming from those zones. However, this solution is more complex than meets the eye, Hawkins explains.

Defining Carbon Offsets

To fight against the negative environmental effects and PR of carbon emissions, large companies like Amazon purchase carbon offsets. A carbon offset is essentially a compensation. When a company cannot physically reduce emissions in one area, it funds the reduction of emissions in another area, to offset the first.

Offsets come in two flavours — mandatory and voluntary. Hawkins tells us mandatory offsets (also known as compliance offset) are purchased because of a legally binding limit to the amount of greenhouse gases that an organisation can release.

On the other hand, individuals and organisations may purchase voluntary offsets at their discretion. As of 2019, the mandatory offset market was $44 billion, while the voluntary offset market was around $300 million.

One issue with carbon offset purchases is that there is no globally agreed upon standard price for carbon offsets. Hawkins states that it's often the case that the cost of buying an offset is much cheaper than the cost of the associated climate damage. In addition to this, voluntary offsets aren't federally regulated, so individuals are forced to perform their own research and make their own assessments in terms of how much they should be paying to offset their emissions.

As a prime example, Hawkins reveals that the Vatican was once presented with offset certificates to make their governing body carbon-neutral — but the millions of trees promised were never actually planted.

The other obvious downside to offsets is that they provide an avenue for wealthy companies to sidestep actually improving or changing their processes, by paying for an offset project elsewhere.

Another concern with carbon offsets is when the calculations should stop.

The relationships are so complex and opaque that it quickly becomes impossible to come up with a straightforward number.

As a developer, when you create a button, have we ever thought about all of the infrastructures behind creating that one button? How do you calculate its carbon footprint? Does the production of your computer's parts count towards the carbon footprint of your button? Should the fossil fuels burned to power your laptop fold into your calculations when deciding how many carbon offsets to purchase? Not to mention the energy used by data centres to store your code.

> Calculating a carbon footprint is a complex and multi-layered beast. The more transparent corporations are about their missions, the easier it will

What are the industries doing about it?

In the US, all tech giants communicate publicly about the rising share of renewable energy sources in their energy consumption, or their goal to be climate-neutral in the near term.

In December 2020, Amazon became the world's largest corporate purchaser of renewable energy, with 187 solar and wind projects across the globe, and projects to power all its operations with renewable energy by 2025.

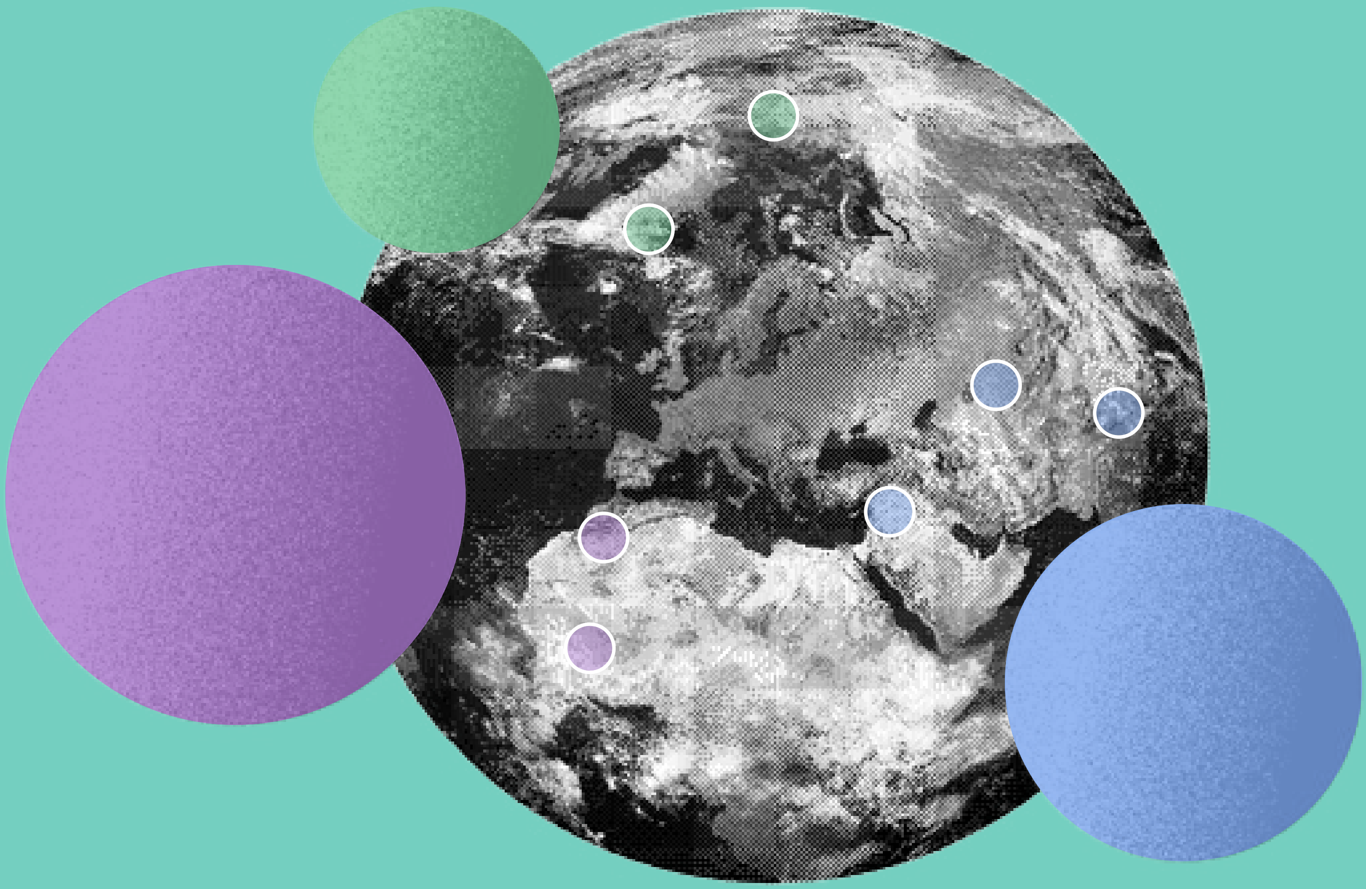
Google has been a carbon neutral company since 2007 and claims that its data centres use roughly half the energy of a typical data centre. Being green is essential to please both investors and customers and has become a central part of corporate strategy – as shown by the listing of the top ten green software companies.

Unsurprisingly, many global initiatives to promote sustainable software are launched by the private sector.

Recently, Microsoft has initiated the Green Software Foundation, together with the Linux Foundation. The goal is to build an ecosystem of people, standards, tooling, and practices to reduce carbon emissions caused by software development. One of their activities will be to develop a certification scheme and training courses for green software developers. Another field of action concerns the development of standards for the software industry as a whole. The declared target is to reduce greenhouse gas emissions by 45 percent by 2020, in accordance with the Paris Climate Agreement.

Stripe, Alphabet, Meta, Shopify, and McKinsey have also launched a new carbon removal initiative worth $925 million.

The initiative — an Advance Market Commitment (AMC) called Frontier — aims to speed up the development of carbon removal technologies by guaranteeing demand for them in the future The idea is to assure researchers, entrepreneurs and investors that there will be a strong ongoing market for these technologies.



What can we do about it?

Green Hosting

According to Hawkins, the first step to creating change is learning whether company data is stored in a carbon-neutral zone or a zone contributing to greenhouse gas emissions.

The Green Web Foundation is a resource that shows what websites (including your own) are hosted by a certified green company. Migrating your company's data to a carbon-neutral region is a big change, but it doesn't take too much effort to start a conversation about it and explore what a migration would require.

As part of our company's For Good strategy and in line with our Climate Active carbon neutral certification Fuller decided to make the first move on behalf of our clients. We approached our host company, Digitize, about a more sustainable hosting option, which they were very interested in exploring.

"At Digitize we agree with Fuller that a more sustainable approach has to be taken with regards to hosting infrastructure in an environmentally friendly way," said Digitize Owner Joseph Mullins.

Our search for a green hosting partner with Digitize has so far resulted in uncovering several businesses who have been greenwashing this offer - further proof of the importance of carrying out due diligence when searching for green solutions. Peak bodies such as the Green Software Foundation and Climate Active have listings of accredited businesses that have verified carbon neutral credentials.

Green Development

There are a few practical tips and tricks that software developers can implement to help lower the carbon footprint of their work, especially when it comes to coding principals.

Using faster code

Many software developers write code to sustain future technologies that run faster and more effectively. Instead of writing code for such hardware, software developers can optimise their code for present hardware, rather than contributing to the requirement for better hardware - let us not forget that the original Apollo 11 Guidance Computer source code for the command and lunar modules came in at just under 4MB.

Consider environmental costs

Throwing more servers at a problem is a financially and environmentally irresponsible way of solving problems - developers need to stop and think of a greener and leaner way.

Storage

Many software developers are looking for the best machinery that holds high volumes of data storage. Rather than looking for these options, remember that a lot of computers currently exist and are lying around with less storage that is still valuable in terms of RAM. Before purchasing the latest computer with the best storage, consider more cost-effective options that do not consume as much energy.

Maintenance and testing

Code, like anything else, is a victim to the march of time. It needs to be revisited and maintained often, to make sure it is working optimally. This practice is referred to as refactoring. When refactoring, the aim should be to improve the readability, performance and energy efficieny of said code, ideally also adhering to the latest best practices too.

Minimise the use of third-party components

Whilst third-party applications can be important for software developers, ensure that whatever is selected to assist a system does not consume more memory and data than is needed. Software developers can assess whether these components are worth it by analysing whether their use case outweighs memory usage.

For example, some plugins that we use fairly regularly that have a negative impact (see: memory and pagespeed) on site performance are:

- The Events Calendar
- Ninja Forms
- Popup Builder
- Popup Maker
- Query Monitor
- Site Kit by Google
- WooCommerce
- Wordfence
- WP Forms
- Yoast
- Contact Form 7
- Redirection

Carbon Offsetting

If green hosting or green development are not options, we can still attempt to offset our emissions by using a service like Cloverly.

Cloverly helps organisations go carbon neutral or carbon negative. Their powerful API calculates carbon emissions and purchases offsets and Renewable Energy Credits in real time to mitigate the environmental impacts of everyday activities.

A plugin called Carbon Offset, developed by Ari Stathopoulos calculates the greenhouse emissions from your website visits and integrates with Cloverly for offsets and payments.

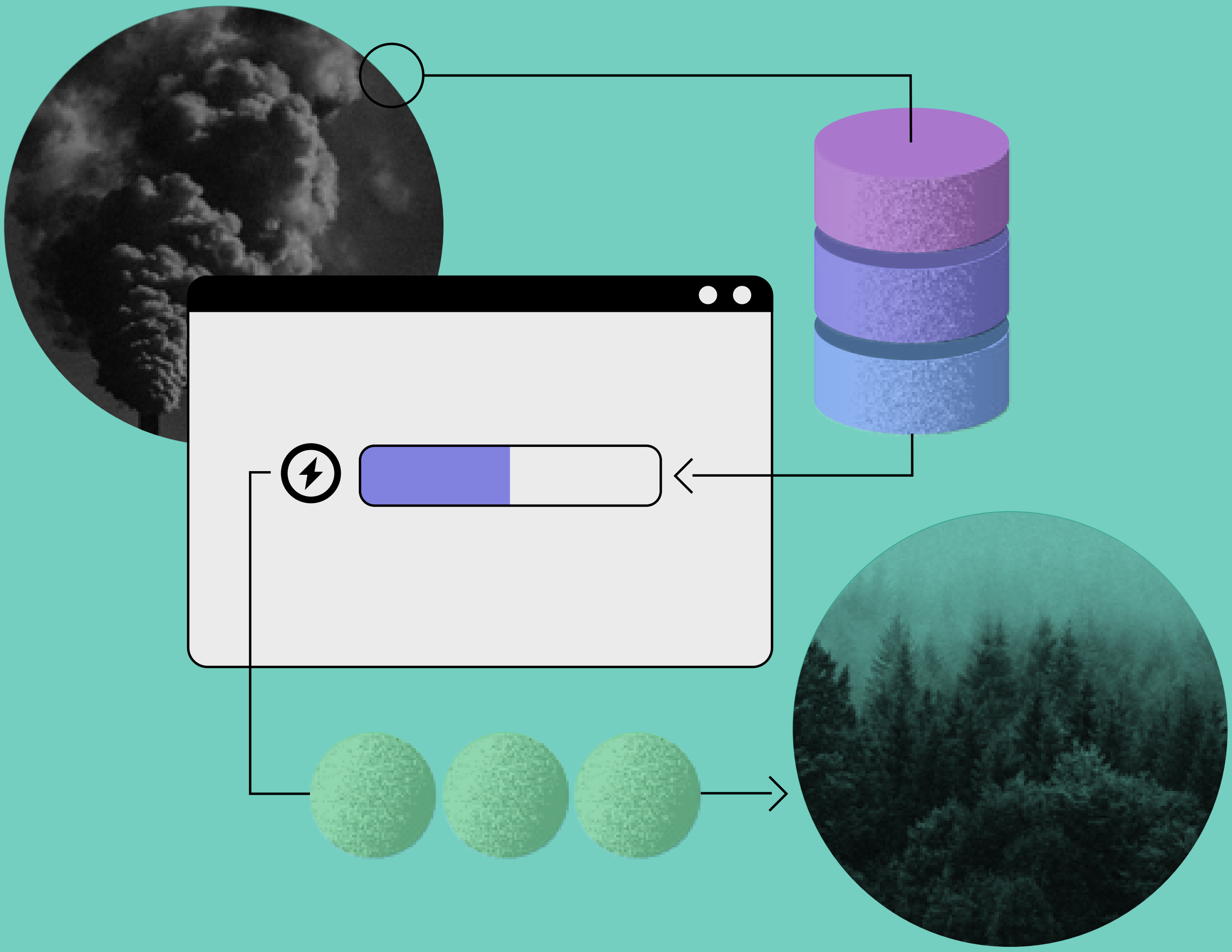
Gooding conveys Ari's view on the issue.

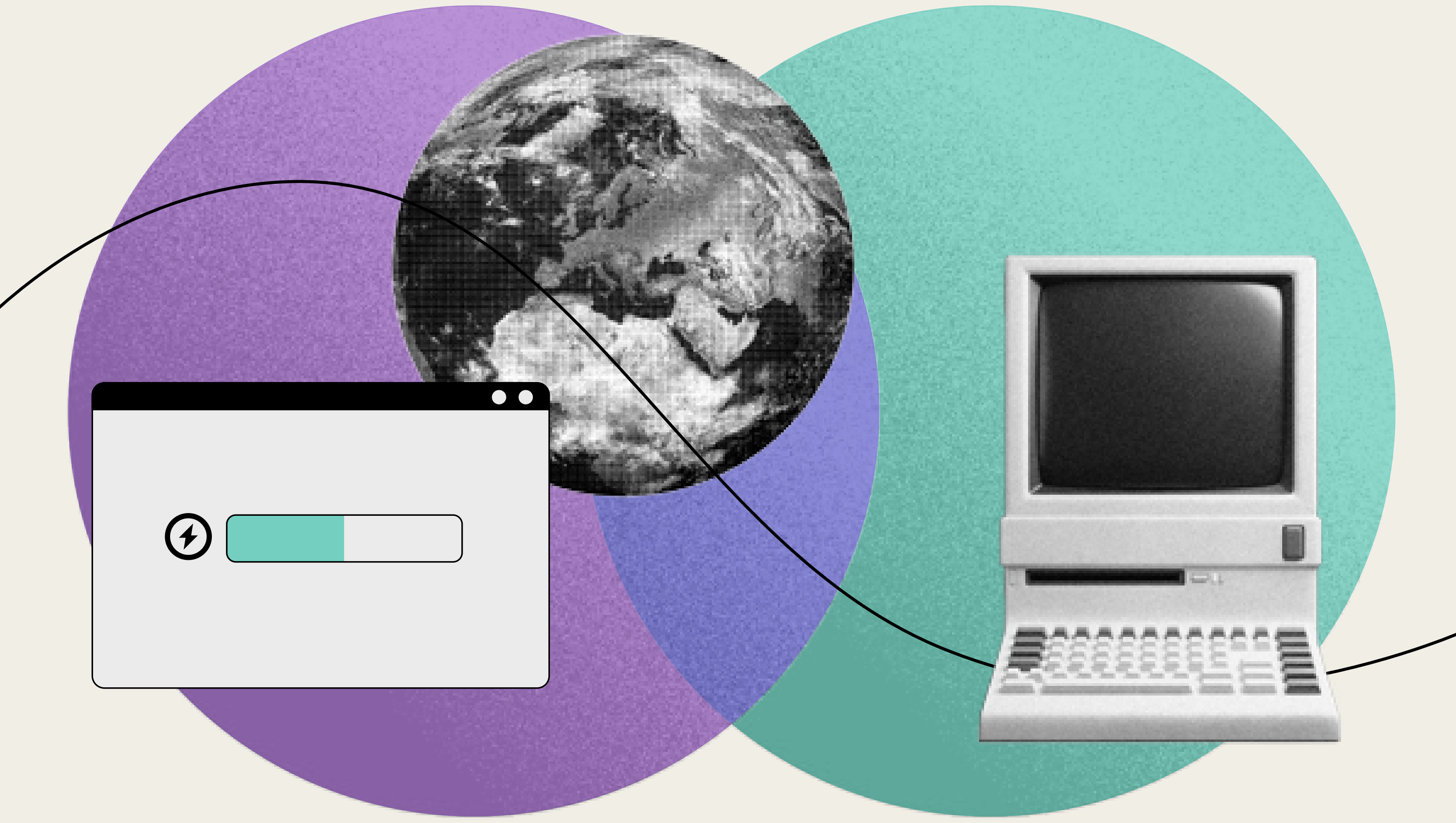
"With WordPress powering 30 percent+ of the web, we're talking about millions of daily views," he said.

"In the unlikely optimistic scenario that all of them generate no more than 0.5g per page-load, WP sites generate no less than 500 metric tons of carbon/day.

"This has nothing to do with WordPress. Instead it's about the 5MB image that the user wants on their frontpage, the fancy wiggling JS animation that requires that extra 5 kb of JS, developers insisting on using jQuery in their themes and plugins, the unused 300kb of CSS that a site has, the Facebook widget, social sharing buttons that use 100kb of JS, or the horrendous use of images of text instead of plain text."

Ari goes on to add that "it's all data that gets downloaded every single time and each time it does, the server runs a few milliseconds more, the browser takes a few more milliseconds to render. It all adds up to wasted energy, energy that took real resources to generate and in the process of doing that, it generated some more carbon emissions."





The Principles

Green Software Engineering is an emerging discipline at the intersection of climate science, software practices and architecture, electricity markets, hardware and data centre design.

The Principles of Green Software Engineering are a core set of competencies needed to define, build and run green sustainable software applications.

These eight principles form a shared understanding of what it means to be a Green Software Engineer independent of application domain, industry, organisation size or type, cloud vendor or self-hosted and programming language or framework.

Through the synthesis of this knowledge, a Green Software Engineer can make decisions which have a meaningful impact on the carbon pollution of their applications.

Who should read this? Anyone building, deploying or managing applications.

1. Carbon

Build applications that are carbon efficient

Greenhouse gases (GHG) act as a blanket increasing the temperature on the surface of the Earth. This is a natural phenomenon, however due to man-made carbon pollution the global climate is changing much faster than animals and plants can adapt. How human society will adapt is still an open question.

There are many different GHGs. The most common is carbon dioxide (CO2). To make calculations easier we normalise all GHG numbers to carbon dioxide equivalent (CO2eq). For example, 1 ton of methane has the same warming effect as about 80 tons of CO2, so we normalise it to 80 tons CO2eq. We may shorten even further to just carbon, which is a term often used to refer to all GHGs.

The goal set by the UN IPCC, and agreed and ratified by 195 states in the Paris Climate Agreement, is to reduce carbon pollution so that the temperature increase stabilises to a 1.5 °C increase by 2100.

The temperature increase on the Earth is dependent on the total amount of carbon we have in the atmosphere, not the rate at which we are emitting. To completely halt the rate of temperature increase, we need to stop adding carbon to the atmosphere or achieve net-zero emissions.

Net-zero means for each gram of carbon we emit we also extract 1 gram, so the overall mass of carbon in the atmosphere remains fixed.

In order to achieve this, we need to mobilise as a global community. We need to start immediately reducing our carbon emissions with the goal of a 45 percent reduction by 2030 and to reach net-zero by 2050.

2. Electricity

Build applications that are energy efficient

Most electricity is still produced through the burning of fossil fuels and is responsible for 49 percent of the carbon emitted into the atmosphere.

All software consumes electricity in its execution. From the applications running on your smartphone to the training of machine learning models running in data centres. One of the best ways we can reduce electricity consumption and the subsequent emissions of carbon pollution made by our software is to make our applications more energy efficient.

The creators of software often do not have to bear the burden of the electricity their software consumes. This is what economists call an externality, i.e. someone else’s problem. A sustainable application takes responsibility for electricity it consumes and is architected to consume as little as possible.

Energy is a measure of an amount of electricity used. The standard unit for Energy is Joules or J, however another common way of referring to energy consumption is in kilowatt-hours or kWh. Throughout the rest of this document we will be using kWh.

3. Carbon Intensity

Consume electricity with the lowest carbon intensity

The carbon intensity of electricity is a measure of how much carbon (CO2eq) emissions are produced per kilowatt-hour of electricity consumed.

The standard unit of carbon intensity is gCO2eq/kWh, or grams of carbon per kilowatt-hour.

Not all electricity is produced in the same way. In different locations and at different times, the electricity is produced using a variety of sources with different carbon emissions. Some sources, such as wind, solar, or hydroelectric, are clean, renewable sources that emit no carbon. Other fossil fuel sources emit varying amounts of carbon to produce electricity. For example, gas-burning power plants emit less carbon than coal-burning power plants.

If your computer is plugged directly into a wind farm, then the electricity it consumes would have a carbon intensity of 0 gCO2eq/kWh, since a wind farm emits no carbon to produce that electricity. Most people can't plug directly into wind farms, they instead plug into power grids that are usually supplied with electricity from a mix of sources that produce varying amounts of carbon. Therefore, when plugged into a grid the carbon intensity is usually a number greater than 0.

Variability of Carbon Intensity

Carbon intensity changes by location since some regions have an energy mix that contains more sources of clean energy than other regions.

Carbon intensity also changes over time due to the variable nature of renewable energy. For example, when it's cloudy or the wind isn't blowing, carbon intensity increases since more of the electricity in your mix is coming from sources that emit carbon.

Carbon intensity changes over time as renewable sources increase or decrease.

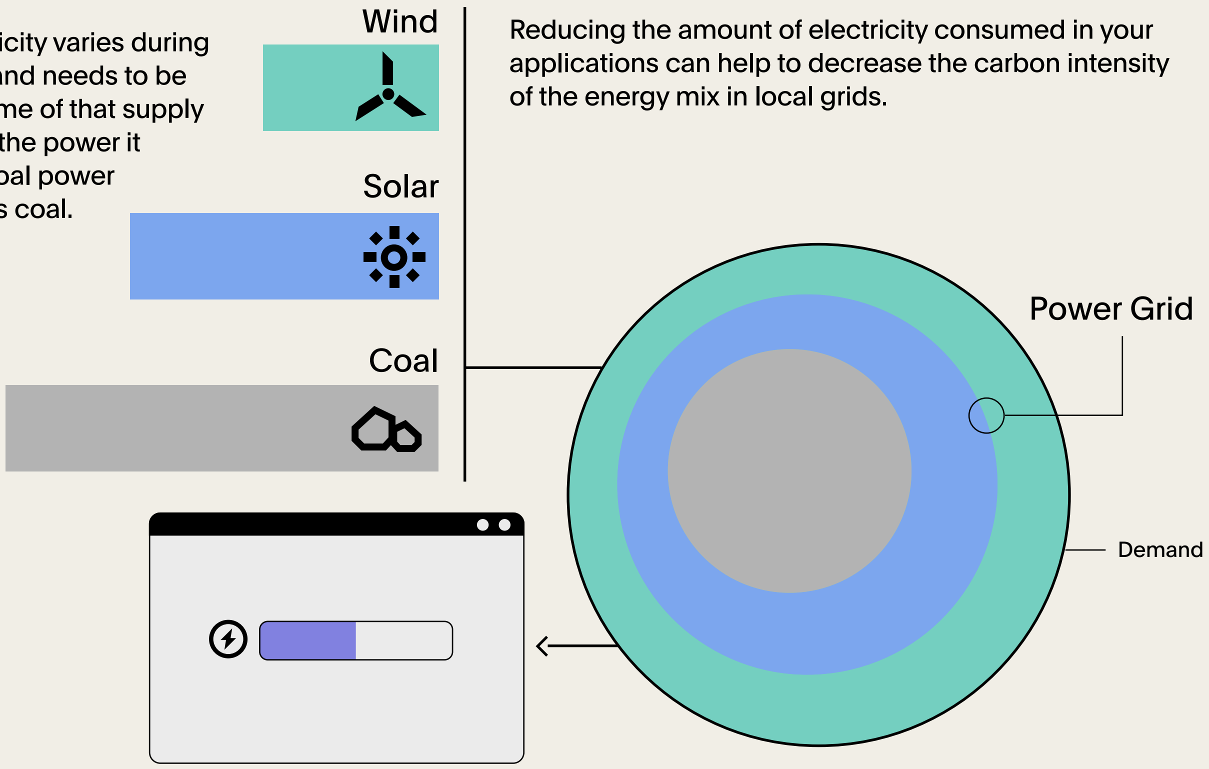
Demand for electricity varies during the day, that demand needs to be met by supply. Some of that supply can easily control the power it produces, e.g. a coal power plant can burn less coal.

Some of that supply can't easily control the power it produces, e.g. a wind farm can't control how much the wind blows, it can only throw away (curtail) electricity that was made essentially for free.

Fossil Fuel sources of power are usually scaled back first and renewables scaled back last.

As a by-product of the way energy markets work as demand for electricity goes down, usually the high emitting fossil fuel sources of power are scaled back first with renewables scaled back last.

Reducing the amount of electricity consumed in your applications can help to decrease the carbon intensity of the energy mix in local grids.

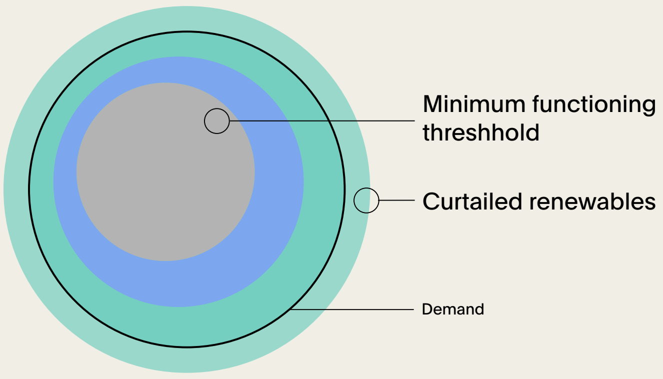


Marginal Carbon Intensity

If you choose to consume more energy, that energy comes from the marginal power plant. The marginal power plant can control the energy it outputs. Renewables cannot control the sun or the wind so marginal power plants are often powered by fossil fuels.

The marginal plant emits carbon. At any moment we have the carbon intensity of both the energy mix in the grid, and the energy that would have to be brought online to meet new demand. This is called the marginal carbon intensity.

Fossil fueled power plants rarely scale down to 0. They have a minimum functioning threshold, and some don't scale at all — they are considered consistent always-on baseload. Because of this, we can sometimes reach the perverse scenario where we throw away (curtail) renewable energy that was created for free in order to consume energy from fossil fuel power plants created with a fuel that costs money.



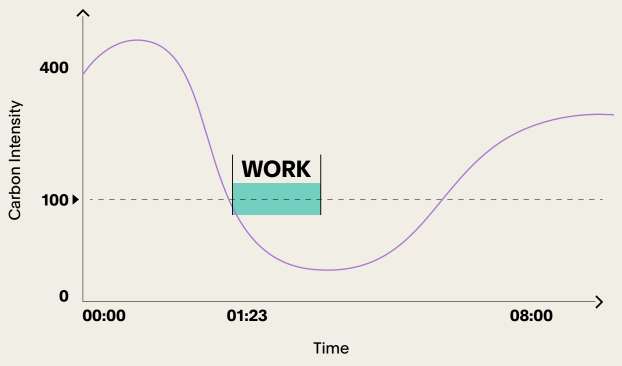
There are moments when the marginal carbon intensity reaches 0.

The marginal carbon intensity could be 0 gCO2eq/kWh when new load would be met with supply from a renewable source that would otherwise have been curtailed.

Demand Shifting

There is currently little in the way of storage or buffering in electrical grid systems. Normally electricity is produced so supply always meets demand. If more energy is being generated from renewables than is needed to support demand, and all our storage options are full, then we curtail (throw away) that clean energy.

One solution is to shift workloads to times and locations where there is more supply of renewable energy. This is called demand shifting.



If you can be flexible with when and where you run workloads then you can then choose to consume electricity when the carbon intensity is less and pause when carbon intensity is high. For example, training a machine learning model at a different time or region where the carbon intensity is much lower.

Studies have shown that these actions can result in a carbon reduction of as much as 45 percent to 99 percent depending on the number of renewables powering the grid.

Look at your application end-to-end, identify opportunities for being flexible regarding workloads and use the carbon intensity of electricity as a signal for when or if to run those workloads.

In this example the purple line is the carbon intensity of electricity. If we shift a workload a little into the future from its preferred start time of midnight, we can take advantage of lower carbon intensity electricity.

Calculating Carbon Intensity

There are several services available which allow you to obtain real-time data regarding the current carbon intensity of different electricity grids, some provide estimates of future carbon intensity, some provide the marginal carbon intensity.

Carbon Intensity API: Free resource for carbon intensity data in the UK.

ElectricityMap: Free for non-commercial single country use, premium solutions for commercial and multi-country access.

WattTime: Free for a single grid region, premium solutions for multi-grid and real-time marginal emissions.

4. Embodied Carbon

Build applications that are hardware efficient

The device you are reading this document from releases some carbon in its creation. Once it reaches the end of life, disposing of it may release more. Embodied carbon (otherwise referred to as "Embedded Carbon") is the amount of carbon pollution emitted during the creation and disposal of a device. When calculating the total carbon pollution for the computers running your software, you need to account for both the carbon pollution to run the computer and the embodied carbon of the computer.

Depending on the carbon intensity of your energy mix the embodied carbon cost of a device can be significant compared to the carbon cost of the electricity powering it.

For example, a 2019 R640 Dell Server has an amortised embedded carbon cost of 320 kg CO_2eq/year. It's also expected to consume 1760.3 kWh/year. The average carbon intensity in the EU is 0.276 kg CO_2eq/kWh.

Therefore the total carbon cost is going to be 320 + (0.276 * 1760.3) = 805 kg of carbon/year of which 320 kg or about 40 percent is from the embodied carbon. Embodied carbon is a significant contributor to the total emitted carbon of hardware.

By thinking of embodied carbon in this way, any device, even one that is not consuming electricity, is effectively releasing carbon over its lifetime.

Don't waste hardware

By the time you buy a computer, it's already emitted a whole load of carbon. They also have an expiry date, computers get old, can't handle modern workloads, and need to be refreshed. If you think about it this way, hardware is then a proxy for carbon, so as Green Software Engineer, we must be hardware-efficient if our goal is to be carbon-efficient.

You can do many things to be hardware efficient, but one thing you can do is help extend the expiry date on hardware. Computers don't wear out, there are no moving parts, they just become obsolete. They become obsolete because we are continually creating software that pushes limits.

Extending the lifespan of hardware

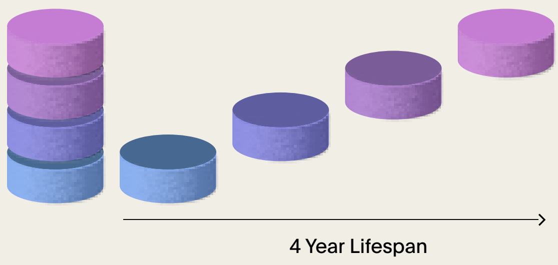
A way to account for embodied carbon is to amortise the carbon over the expected life span of a device. For example, if it took 4 tons of carbon to build a server and we expect the server to have a 4-year lifespan, we can consider this equivalent to 1 ton of carbon being released per year during its lifespan.

If we just added one more year to the lifespan of our 2019 R640 Dell Server then the amortised carbon drops from 320kg CO2eq/year to 256 kg CO2eq/year.

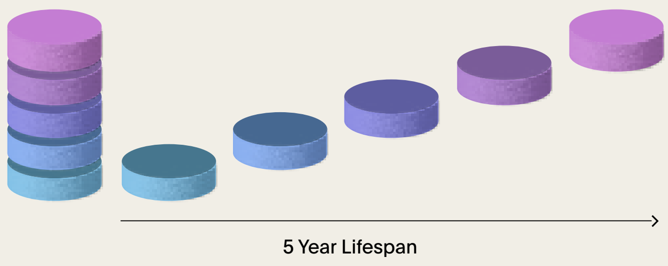
Hardware is retired either because it breaks down or because it struggles to handle modern workloads.z

Software cannot help with the first however if we focus on building applications that can run on older hardware, we can help with the second.

Embodied carbon of a server amortised over 4 years.



Embodied carbon of the same server amortised over 5 years.



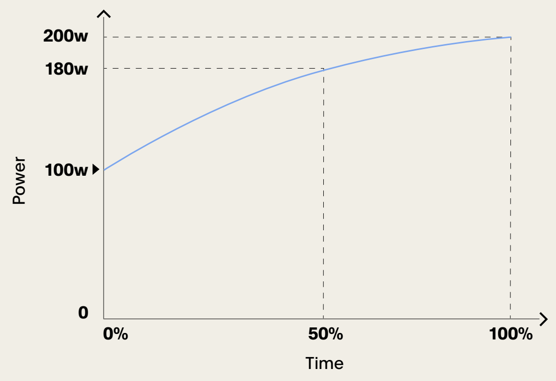
5. Energy Proportionality

Maximise the energy efficiency of hardware

Energy proportionality is a measure of the relationship between power consumed in a computer system and the rate at which useful work is done (its utilisation).

Utilisation is a measure of how much of a computer's resources are being used, usually given as a percent. An idle computer has a low utilisation percentage and isn't being utilised. A computer running at its maximum capacity has a high percentage and is being fully utilised.

The relationship between power and utilisation is not exactly proportional.



At 0 percent utilisation the computer still draws 100W, at 50 percent utilisation it draws 180W and at 100 percent utilisation it draws 200W. The relationship between power consumption and utilisation is not linear and it doesn't cross the origin.

Because of this, the more you utilise a computer, the more efficient it becomes at converting electricity to useful computing operations. Running your work on as few servers as possible with the highest utilisation rate maximises their energy efficiency.

An idle computer, even one at zero percent utilisation, still draws electricity. This static power draw varies by configuration and by hardware components, but all components have some static power draw. This is one of the reasons PCs, laptops, and mobile devices have power-save modes available. If the device is idle it will eventually trigger a hibernation mode and put the disk and screen to sleep or even change the frequency of the CPU. These power-save modes save on electricity, but they have other trade-offs, such as a slower restart when the device wakes up.

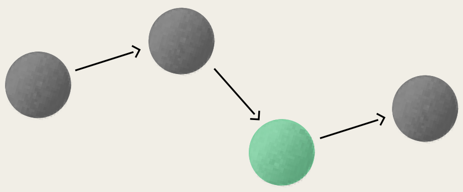
Servers are usually not configured for aggressive or even minimal power-saving. Many server use-cases demand full capacity as quickly as possible in response

to rapidly changing demands. This can leave many servers in idle modes during low demand periods. An idle server has a cost both from embedded carbon and its inefficient utilisation.

The most efficient and green approach is to run your work on as few servers as possible with the highest rate of utilisation.

6. Networking

Reduce the amount of data and distance it must travel across the network



A network is a series of switches, routers, and servers. All the computers and network equipment in a network consume electricity and have embedded carbon. The internet is a global network of devices typically run off the standard local grid energy mix or powered by renewables.

When you send data across the internet, you are sending that data through many devices in the network, each one of those devices consuming electricity. As a result, any data you send or receive over the internet emits carbon.

Nodes in a network run on different energy mixes

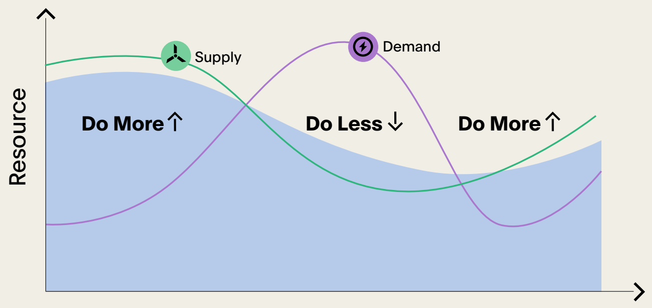
The amount of carbon emitted to send data depends on many factors including:

- Distance the data travels
- The number of hops between network devices
- The energy efficiency of the network devices
- The carbon intensity of energy in the region of each device at the time the data is transmitted.
- The network protocol used to coordinate data transmission - e.g. multiplex, header compression, TLS/Quic

7. Demand Shaping

Build carbon-aware applications.

Demand shifting is the strategy of moving workloads to regions or times when the carbon intensity is less, or to put it another way when the supply of renewable electricity is high.



Demand shaping is a similar strategy, but instead of moving demand to a different region or time, we shape our demand, so it matches the existing supply.

If supply is high, increase the demand - do more in your applications - if the supply is low, decrease demand - do less in your applications.

A great example of this is video conferencing software. Rather than streaming at the highest quality possible at all times, they often shape the demand by reducing the video quality to prioritise audio.

Another example is TCP/IP. The transfer speed ramps up in response to how much data can broadcast over the wire.

A third example is progressive enhancement with the web. The web experience improves depending on the resources and bandwidth available on the end-users device. Microsoft have produced a guide for how you can measure the power consumption of your frontend.

Carbon-aware vs. carbon-efficient

Carbon efficiency can be transparent to the end-user. You can be more efficient at every level in converting carbon to useful functionality while still keeping the user experience the same.

But at some point, being transparently more carbon-efficient isn't enough. If the carbon cost of running an application right now is too high, we can change the user experience to reduce carbon emissions further.

At the point the user is aware the application is running differently, it becomes a carbon-aware application.

Demand shaping carbon-aware applications is all about the supply of carbon. When the carbon cost of running your application becomes high, shape the demand to match the supply of carbon. This can happen automatically, or the user can make a choice.

Eco-modes

Eco-modes are often used in life: for instance in cars or washing machines. When switched on, the performance changes as they consume fewer resources (gas/electricity) to perform the same task. It's not cost-free (otherwise, we would always choose eco-modes), so we make trade-offs. Because it's a trade-off, eco-modes are almost always presented to a user as a choice, and the user decides if they want to go with it and accept the compromises.

Software applications can also have eco-modes which when engaged changes application behaviour in potentially two ways:

- Intelligence. Giving users information so they can make informed decisions.
- Automatic. The application automatically makes more aggressive decisions to reduce carbon emissions.

Summary

Demand shaping is related to a broader concept in sustainability, which is to reduce consumption. We can achieve a lot by becoming more efficient with resources, but at some point, we also just need to consume less. As Green Software Engineers, to be carbon-efficient means perhaps when the carbon intensity is high, instead of demand shifting compute, we consider cancelling it. Reducing the demands of our application and the expectations of our end users.

8. Measurement and Optimisation

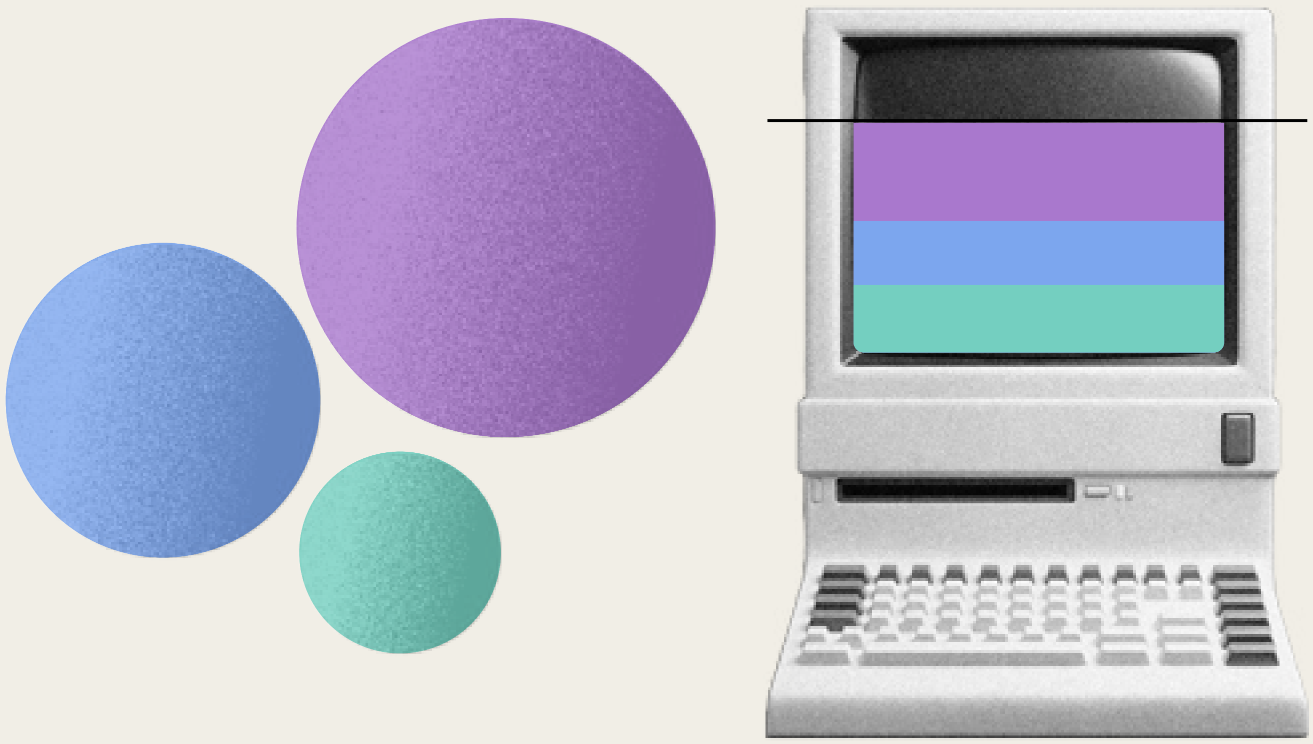
Focus on step-by-step optimisations that increase the overall carbon efficiency

Sustainability isn't one optimisation, it's thousands. One piece of advice is to look end-to-end and take it step by step. Often putting in the effort to understand the full stack, from user experience to data centre design or electricity grids yields simple solutions that significantly improve carbon efficiency.

Weigh up the effort required to decarbonise vs the potential rewards. Just like the broader global sustainability movement, some sectors will be harder to decarbonise than others. In computing, some application domains will be harder to decarbonise than others. Some parts of your application architecture will be harder to decarbonise than others.

The key to success in optimisation is to choose a measurement criterion that will give clear signals as to where best to put optimisation efforts. For example, is it worthwhile to spend two weeks reducing megabytes from network communication if the database queries cause 10 times more carbon to be emitted?

Rarely, can we directly measure our application's carbon cost, but if we follow a resource chain down and it eventually has a link to carbon emissions, then that is a good proxy for carbon.



Carbon

Measuring emitted carbon is a complex challenge, with parts of the stack that need to be estimated rather than measured, but with some effort, it's possible.

Because of the variability of carbon intensity and other dependencies, the total carbon emitted may change depending on the time of day or region the application is run.

The same application measured at different times will result in different amounts of carbon. This could be a good signal, especially if you are open to demand-shifting workloads or it could be noise if you are focussing on energy optimisations.

Energy

The energy consumed by your application may vary every time it runs, this may be something you want to take as an optimisation signal, or this may be something you want to control for.

The same application run on different hardware may result in different amounts of energy consumed because of the differences in energy efficiency between the hardware components.

Because of the energy proportionality principle, the same application runs on the same hardware but at different times may result in different amounts of energy consumed because the utilisation of the hardware is different between the two runs. That is, the hardware might be running other applications during the second run, and this changes the hardware's overall energy efficiency.

Overall, though, creating applications that consume less electricity for the same human-perceptible performance and output is a good proxy for carbon reduction.

There are devices, tools and libraries available that help you measure the energy consumed by an application.

- PowerAPI A system monitoring library only works for GNU/Linux and only calculates CPU energy; it does, however, calculate the energy used per process.
- Intel Power Gadget Only works on Intel Core processors, only calculates power consumption due to the CPU and does not break this out on a per-process basis.
- PowerCFG A Windows 10 tool allowing to have the electrical consumption per process.

A thorough analysis of the various software and hardware tools to measure energy consumption can be found in the paper Software development methodology in a Green IT environment.

Cost

At some point, the cost of electricity is factored into most services. Building applications that run as cheaply as possible is usually a good proxy for applications that emit less carbon.

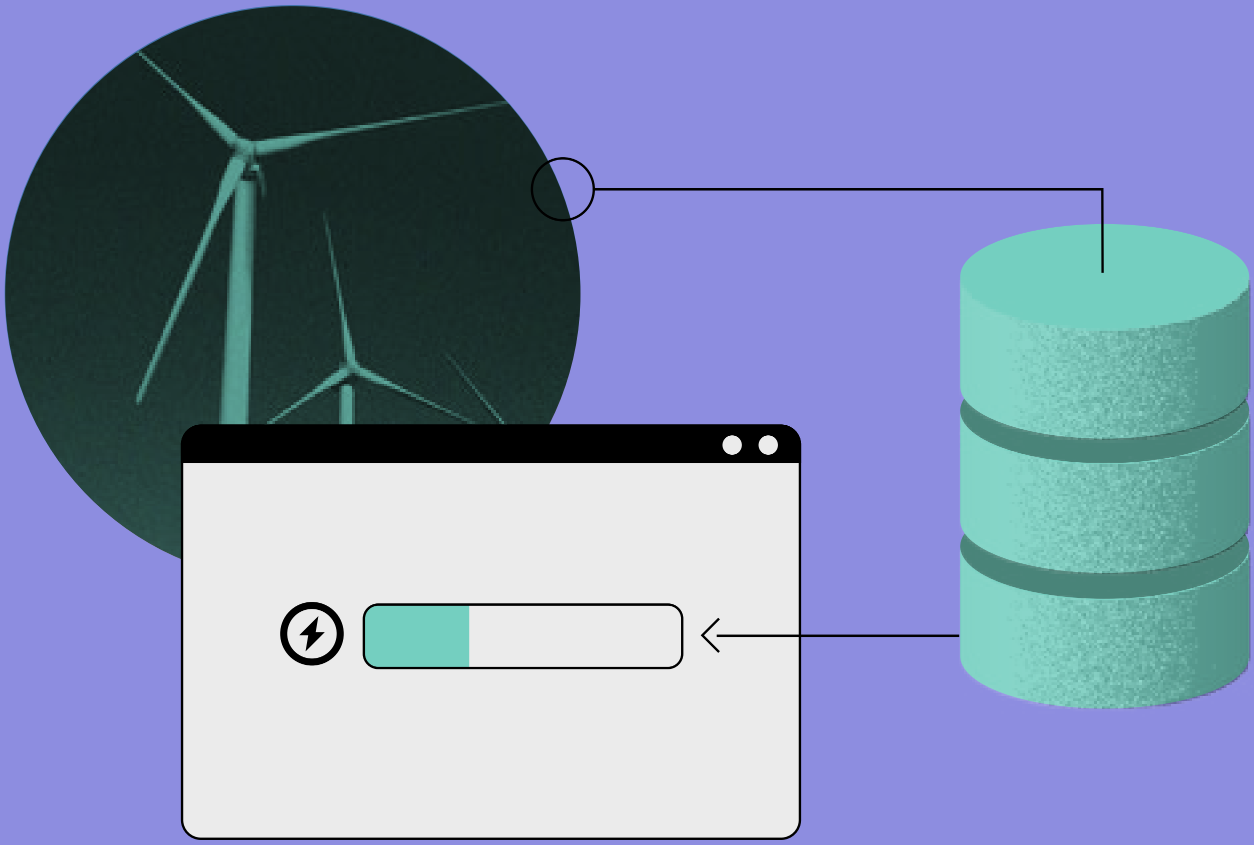
Networking

The cost of electricity in networking is often not considered. The number of services that offer unlimited bandwidth for a single price means there is little price pressure to reduce bandwidth.

Measuring and then reducing the amount and distance your data must travel is a good proxy for reducing carbon.

Performance

If you can architect an application that performs better for the same level of utilisation, then this is likely to reduce overall carbon.



Applying the principles

We've defined the problems, and we've defined the principles that we should be aware of when attempting to alleviate the problems — but how do we actually apply those principles on a daily basis?

Computer Weekly outlines 8 ways to make your applications more energy efficient.

Reduce resolution of images and/or send them less frequently

When pictures are used in mobile applications, these pictures must be sent over the internet and radio connection, which is a major source of energy consumption. By reducing the size of these images or uploading them less often, energy can be saved. In particular, applications that run in a mobile browser can benefit greatly from this quick win.

Run multiple applications on shared servers

If each application runs on its own server, these servers will be doing nothing useful most of the time. But they are on, and consume electricity, all of the time. By letting applications share servers, fewer servers are needed to do the same work. Many applications can be easily made suitable to run on shared servers.

Reduce data translation between components

To make various components of a software system work together, the data that they exchange may need to be translated. Such translations can be computationally expensive. Sometimes, multiple translation steps are performed rather than translating from source to target in a single step. Also, the intermediate formats can be overly verbose, leading to big files being transferred that could be much smaller. By simplifying the formats and the translation steps, unnecessary energy consumption can be reduced.

Log less

While developing software, it is useful to let the application log the steps it is taking and their intermediate results, to diagnose bugs or other problems. Writing logging information to disk and storing large log files consumes energy. But when the finished application is deployed in a production environment, much of this logging information is not used. Employing a logging framework that allows the degree of logging to be reduced when logs are not used means unnecessary energy consumption can be avoided.

Delete historic data

An application that supports a certain business process needs to store information about business transactions. After the business transaction has been concluded, most applications will retain data related to that transaction for possible future reference. This data is often kept ‘alive’ without being used. Year after year, it sits in memory or on a hard disk and thus causes energy consumption. By purging old transactions from the application database, the storage needs can be prevented from growing larger and larger, and energy can be saved.

Compile interpreted languages

Many applications are developed in programming languages that are not compiled to efficient machine code before deployment, but that are translated by an interpreter to machine code while the application runs. This interpreted code typically requires more processing power, and more servers consuming more energy to do the same work. For some of these interpreted languages, such as PHP, a compiler is also available. By making small changes to the application code, it can be made suitable for compilation, after which it can be run more (energy-)efficiently.

Refrain from frivolous features

Modern development toolkits make it possible to create astonishing user interfaces, full of graphics, animations, assisted editing, suggestions for further navigation, and more. Sometimes these features are useful because they enrich the user experience and make the user more productive in the tasks they wish to accomplish. When copying a file from one folder to another, do you really need to watch sheets of paper fly from one end to another? Features that offer no value to the user generally do consume energy, which can be saved by omitting them.

Avoid chatty protocols

The communication protocols between components of an application can involve many messages being sent back and forth. In the case of smartphone apps, radio traffic is a major source of energy consumption, which can be reduced by not establishing a new radio connection for each message, but saving up several messages until a number of them can be sent at once. For example, apps could save information to the server only after a number of data-entry screens have been completed, rather than after completing each screen.

Asim Hussain adds to these suggestions below.

Optimise your network traffic

Reduce the amount of traffic your architecture creates per operation as well as the distance each request and response travels.

Consider using caching headers, which allows browser caches and proxy caches to have enough information to confidently cache static assets. Caching static assets at the browser or proxy level allows future requests for those assets to be handled by those caches and reduces network traffic to your application.

Consider using a CDN to distribute your application's static assets closer to the source of a request. This distribution of assets reduces the distance all requests for static assets have to travel over the network.

Where possible, reduce the size and optimise your bundles and static assets.

Consider using compression and decompression for data you transmit over the network. Compression and decompression usually takes less overall energy than transmitting uncompressed data over the network.

Increase your compute utilisation

Update your workload distribution and compute resources so that you use less resources at a higher utilisation. This reduces the amount of energy your compute resources spend in an idle state, or using energy without doing work.

If using virtual machines for compute resources and they have low utilisation, consider reducing the size of those virtual machines to increase utilisation. Smaller virtual machines with higher utilisation usually use less energy than larger virtual machines with lower utilisation, given the same workload.

Evaluate migrating your workload to a PaaS (Platform as a Service) where possible. Typically, PaaS solutions are sized more appropriately for their workload and can run those workloads at a high utilisation on their underlying compute resources.

Consider using auto-scaling or burst capabilities for your compute resources over statically allocating compute resources for maximum capacity at all times. These capabilities allow you to increase and decrease your compute resources based on demand while keeping the utilisation high on those compute resources.

Optimise your database

Optimising which database you use as well as how the data is stored can reduce the energy used to run the database as well decrease idle time waiting for queries to complete.

- Ensure you are using the best database for interacting with your data set. For example, if you are running many relational queries on your data set, a relational database is better suited and likely more efficient to use than NoSQL database.
- If no single database is designed to handle all the ways you interact with your data set, consider keeping redundant copies of your data in different databases, and using each database for the subset of interactions best suited for that database.
- Consider using an index if your database offers it.
- Consider evaluating and optimising your queries.
- Consider using a database cache. In some cases, caching can reduce redundant queries to the database and decrease energy usage by the database, especially for complex or compute-intensive queries.

Understand your latency limits

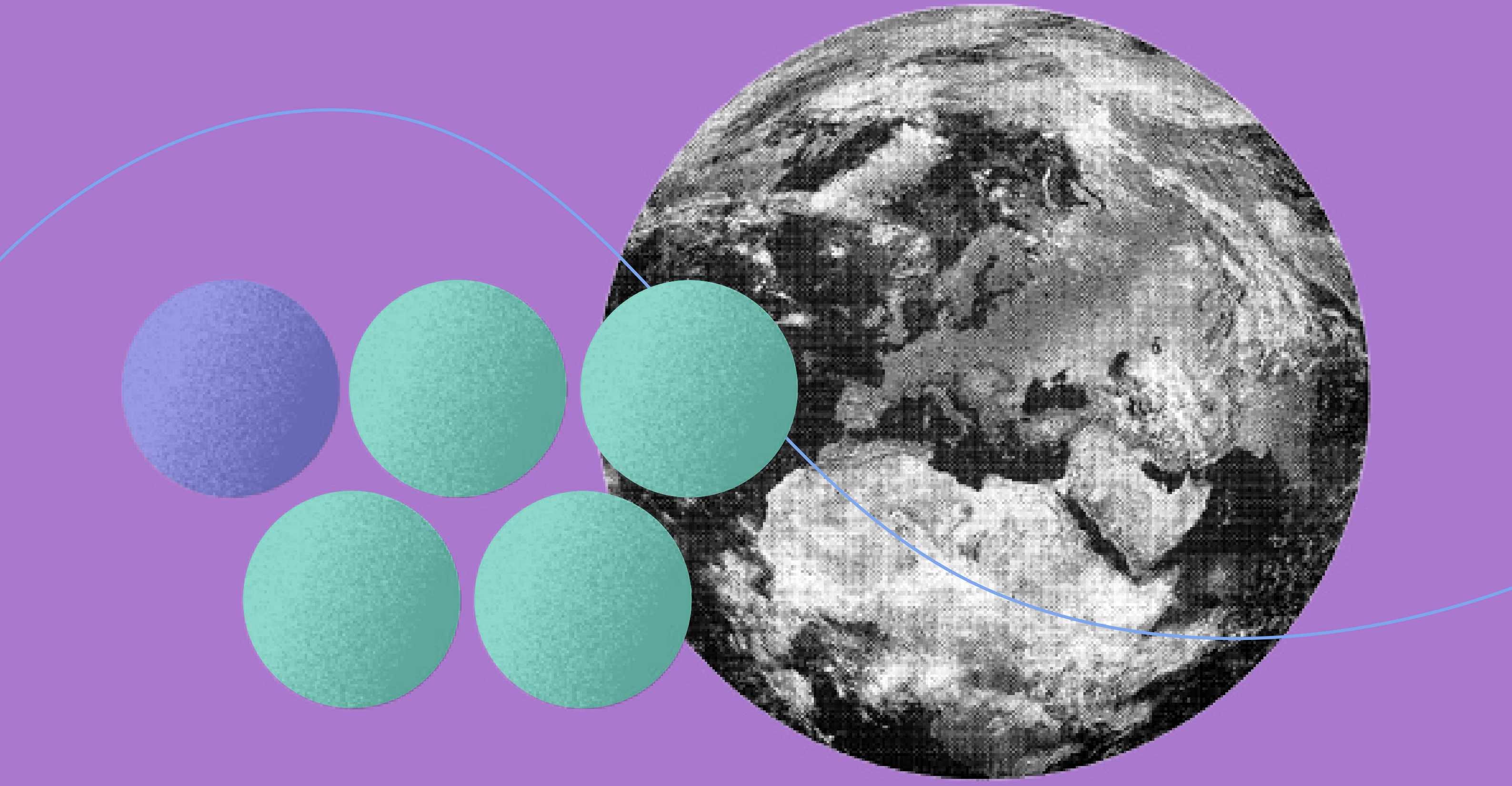
In many cases, web applications are designed by default with very low latency expectations, assuming a response to a request should happen immediately or as soon as possible. This assumption can limit your options for reducing the energy usage in your application. Consider evaluating how your application is used and if you can relax the latency limits in some areas, which can increase your options for reducing carbon.

- Consider separating certain operations outside of the request/response cycle. For example, if there is a request to send an email that blocks the response until the email is sent, you can instead asynchronously send the email using a worker process and unblock the response.
- Consider running worker processes a lower priority than web processes. This prioritisation allows worker processes to run only when compute resources are not needed by web processes and keeps utilisation high.
- Consider running the worker processes in a region with lower carbon intensity.
- Consider delaying the worker process to run when the carbon intensity is the lowest.

Reduce your number of microservices

A microservices architecture is an effective way to focus a service around a specific business domain and decentralise ownership and knowledge throughout the team or system. Ensuring the appropriate level of abstraction is important to help limit network congestion, latency, and overall complexity.

- Consider combining services, logically or physically, where similar scale points exist to reduce the footprint of the overall architecture.
- If two or more microservices are highly coupled, consider co-locating to reduce network congestion and latency.
- Use languages and technology stacks that optimise the efficiency of a specific microservices function. The independence and abstraction of functionality to an API layer means you are free to make the technical decisions that maximise utilisation in your technical stack for each microservice.
- Consider running any resource-intensive microservices in a region with a lower carbon intensity.



Conclusion

As Green Software Engineers, we believe everyone has a part to play in the climate solution.

If you are reading this document and identify as 'green', know you are part of a massive global movement of people who care and are taking action. Sustainability champions work in every discipline across engineering, from designing silicon to designing user experiences.

Whatever sector, industry, role, technology – there is always something you can do to have an impact.

Nothing happens in isolation, everything is connected, and small changes lead to big changes.

Even the act of normalising a discussion about sustainability in technical meetings will empower others to raise their voice. That's how you create change in any organisation. Sustainability is enough, all by itself, to justify our work.

As Green Software Engineers, we recognise there are many advantages to building sustainable applications.

They are almost always cheaper, they are often more performant, they are often more resilient.

But the primary reason we are practising Green Software Engineering is for sustainability – everything else is an added advantage.

References

https://www.mybib.com/j/LeanLegalAnt