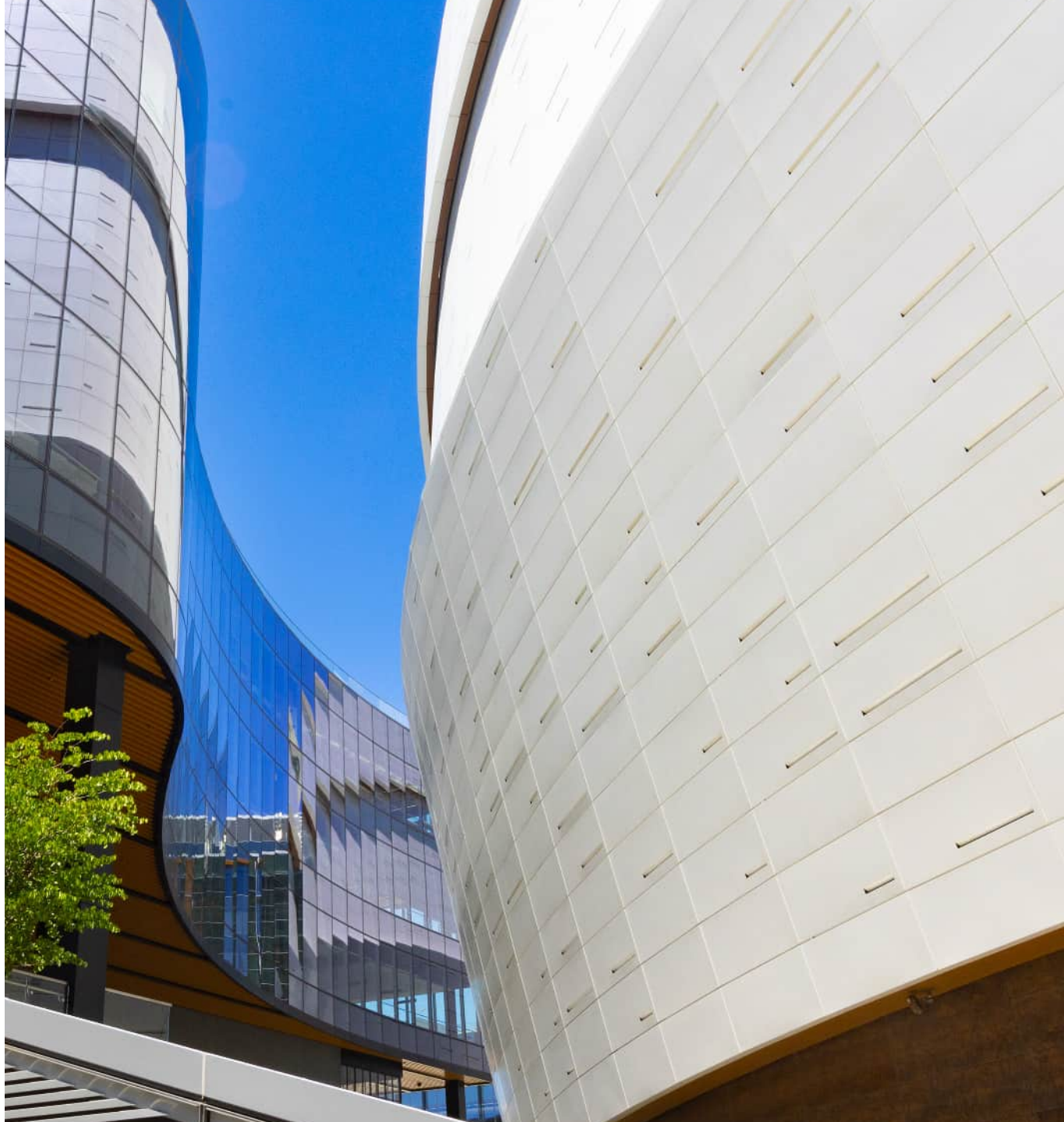
Microsoft

Sustainable Software Development Guidelines

SUPPORTED BY

INFOCOMM MEDIA DEVELOPMENT AUTHORITY



Contents

- Introduction 03
- Sustainable Software Development Principles 06
  - Principle 1: Energy Efficiency 08
  - Principle 2: Carbon Awareness 15
  - Principle 3: Hardware Efficiency 19
- Software Carbon Intensity Standard 22
- Toolkit and Resources 26
- Recommendations for Developers 29
- Appendix 31

01

Introduction

Jeth Lee
Chief Legal Officer, Singapore
Microsoft

Sustainability is a pressing issue that requires the world's attention and ingenuity. Carbon emissions, including emissions across the supply chain from electricity generation to energy needed to power end-point devices, are challenging to reduce and manage.

In view of these challenges, organisations must remain committed to building a more sustainable future and helping their stakeholders, partners and customers to do the same. As the global demand for digital technology continues to grow, so too does the carbon footprint of the software industry. Sustainability efforts span the breadth of reducing emissions through the increased use of renewable energy and in ensuring that software, on which upstream processes are run, are inherently efficient.

A new paradigm of sustainable software has emerged, which lies at the intersection of climate science, software design, clean energy, and digital infrastructure. Sustainable software is carbon-efficient software that aims to minimise its emissions profile by design.

The guidelines has been applied through the Singapore GreenTech Challenge, where local developers were invited to build sustainable and carbon-aware applications that contribute to the pillars of the Singapore Green Plan 2030.

Winners of the Singapore GreenTech Challenge have been selected in recognition of their innovative solutions in setting and tracking sustainability targets, identifying and transitioning to renewable energy sources, and engendering tradeable carbon assets.

This paper will present an overview of sustainable software development principles and how they can be applied to the engineering and development process. We will discuss measurement standards, such as the Software Carbon Intensity (SCI) methodology. In addition, we will introduce a range of tools that developers can use to create more sustainable software, including the Carbon Aware software development kit (SDK) and other tracking and reporting tools, such as the Microsoft Sustainability Manager. Finally, we will provide a list of best practices and recommendations for developers looking to adopt sustainable software principles in their work.

By following this guidelines, developers can reduce carbon emissions from the software they create and make a positive impact on the environment.

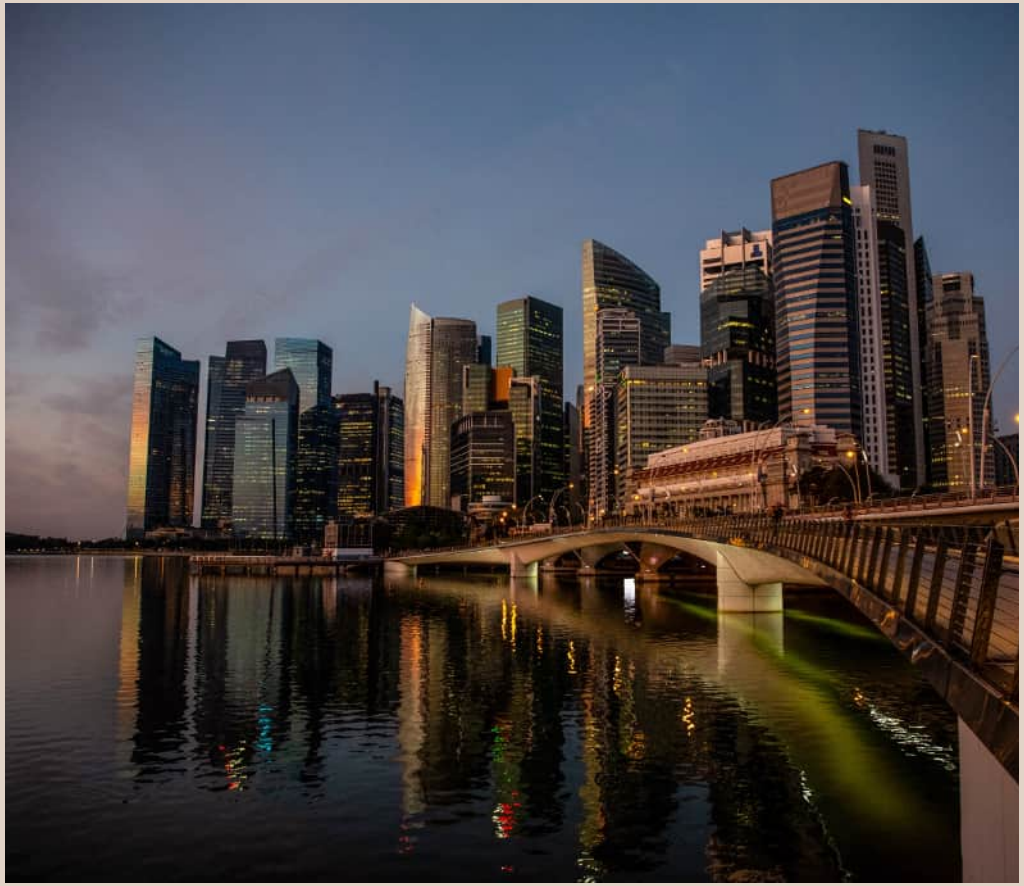
Feature 1:

Singapore advances plans to become a leading digital sustainability hub

Singapore's Infocomm Media Development Authority (IMDA) is partnering with Microsoft to address climate-related issues and improve sustainability outcomes for digital technologies. This partnership seeks to accelerate global and local development of software applications and solutions to help industries do more with less.

IMDA and Microsoft will share learnings, certification pathways, best practices, and standards for effective measurement and reporting of carbon emissions arising from software applications. This partnership will look to advance the implementation of principles and tools for the development of green technologies.

As part of this partnership and through this paper, IMDA and Microsoft have developed joint guidelines that outlines guidance for the development of sustainable software. The guidelines have been applied through the Singapore GreenTech Challenge, where developers built sustainable and carbon-aware applications that will contribute to the energy reset pillar of the Singapore Green Plan 2030.



02

Sustainable Software Development Principles

What is Sustainable Software?

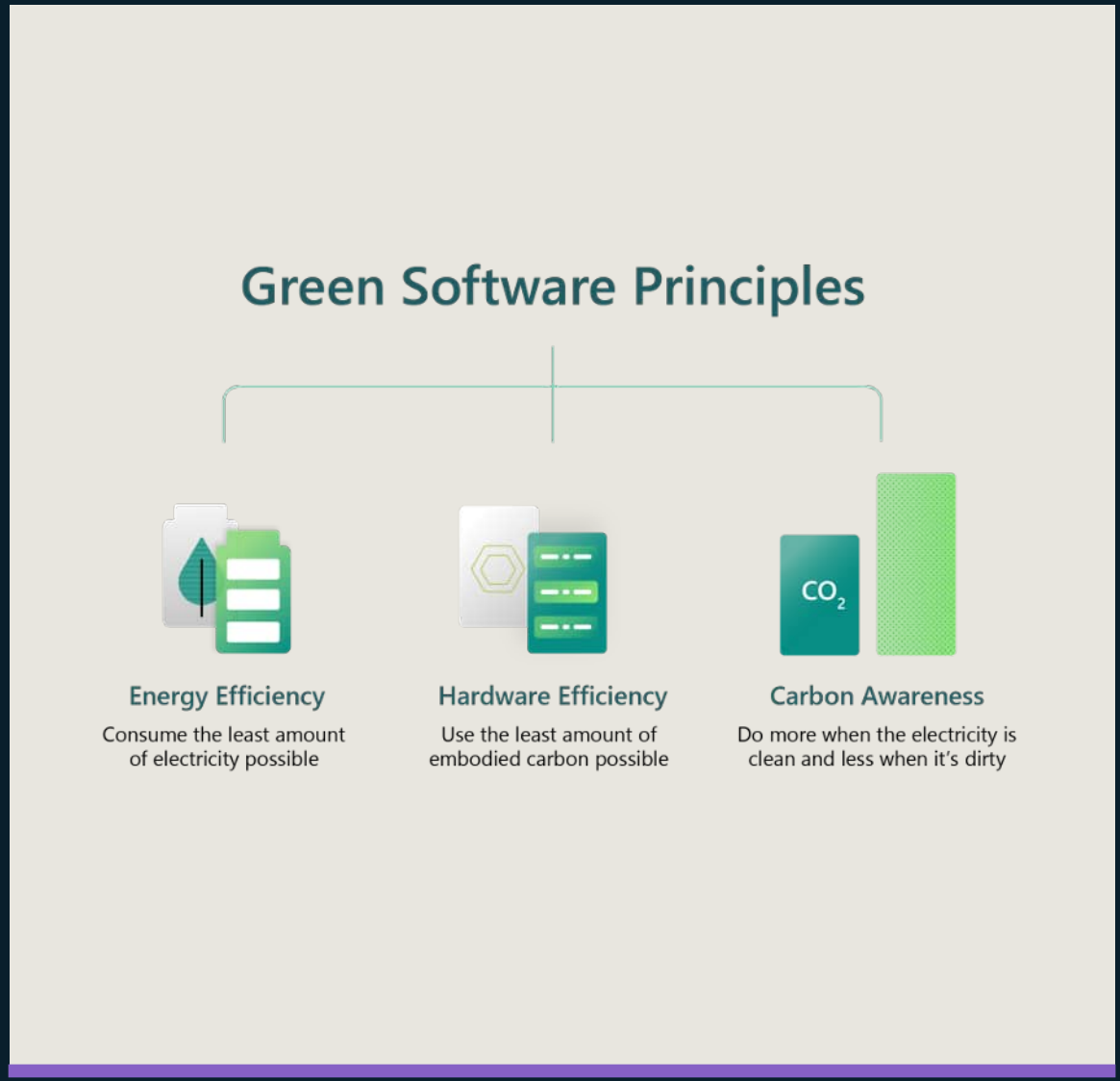
Sustainable software, or green software, is an emerging discipline with principles and competencies to define, develop and run applications with sustainability in mind. It lies at the intersection of climate science, software design, electricity markets, hardware, and data centre design. Sustainable software is carbon-efficient, meaning it emits the least carbon possible by design through code, architecture and other intentional choices.

Sustainable software applications tend to be cheaper to run, more performant, more resilient and more optimised, all while having a positive impact on the planet.

Three main principles guide the reduction of carbon emissions of software:

- **Energy efficiency** – software should use the least amount of energy possible
- **Carbon awareness** – software should do more when electricity is cleaner, and do less when electricity is less clean
- **Hardware efficiency** – the least amount of embodied carbon (i.e. the amount of carbon emitted during the creation and disposal of a device) should be used

This paper will detail certain sustainable software patterns, which are specific examples of how to apply one or more principles in a real-world scenario. While principles describe the overall theory underpinning sustainable software, patterns are practical guidance that practitioners can use in their software applications today. The Green Software Foundation publishes a catalogue of vendor-neutral sustainable software patterns across various categories.



Principle 1:
Energy Efficiency

Greenhouse gases are a group of gases contributing to global warming. Carbon is often used as a broad term to refer to the impact of emissions and activities on global warming, although it is technically just one of the elements involved. CO_2e is a measurement term used to measure this impact.

The first principle of sustainable software development, carbon efficiency, goes towards our goal of emitting the least possible amount of carbon per unit of work.

All software consumes energy. Sustainable software is designed to consume as little energy as possible. We should make sure that, at every step in the process, waste is minimised and most of the energy goes to the next step.

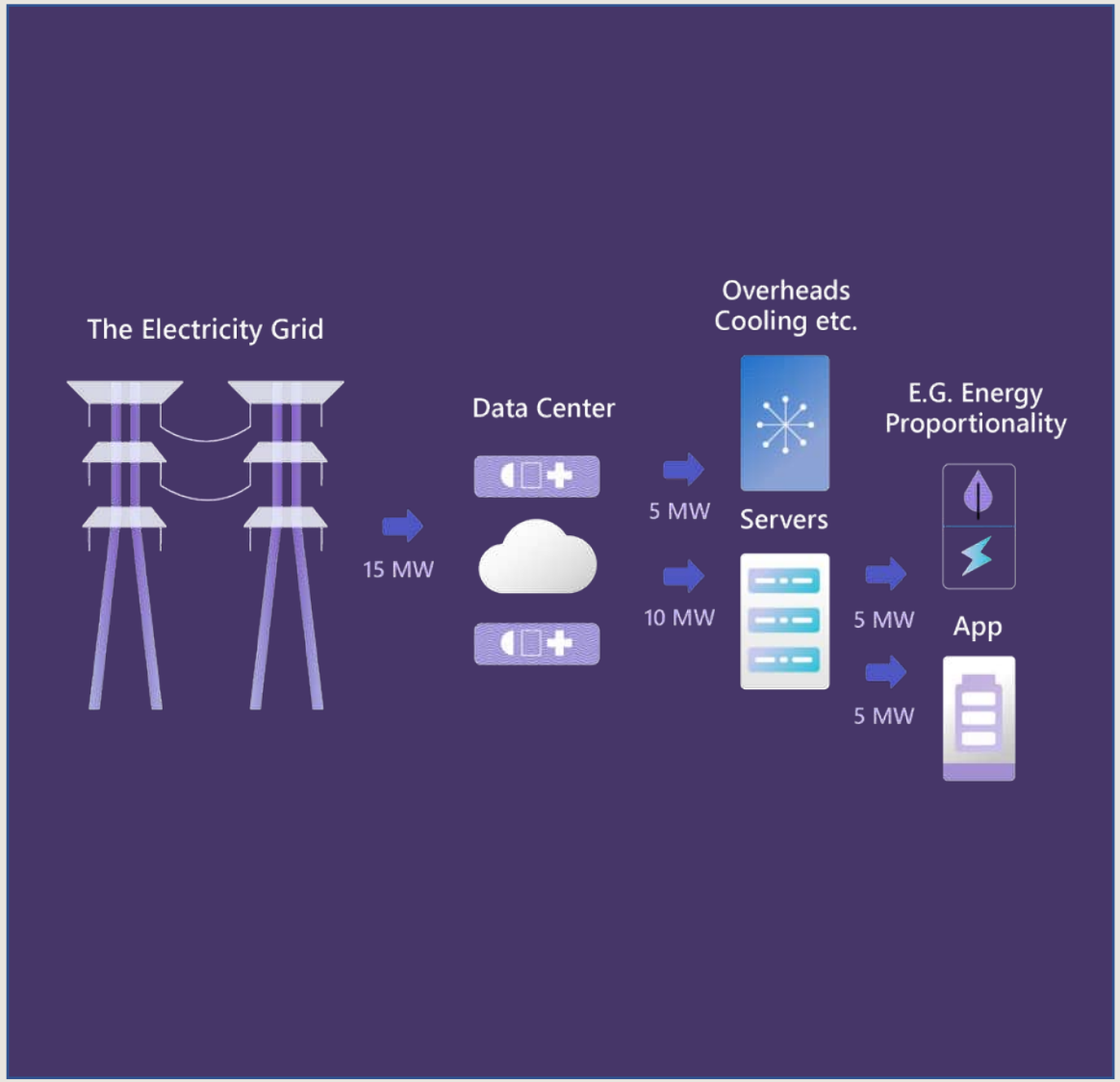
Principle 1:
Energy Efficiency

User interface and experience

The final step in this chain is represented by the end-user of a product. Our goal is not simply to make the most energy efficient code or the "greenest" software, but to design for the end-user and facilitating decision that do not result in unnecessary emissions.

This might mean batching jobs together to take advantage of energy proportionality (the relationship between power consumed by a computer and the rate at which work is done) or changing how a user uses your software.

There are various ways to understand the effectiveness of the energy that is being supplied, which may then influence the manner in which you may wish to design the relevant software.

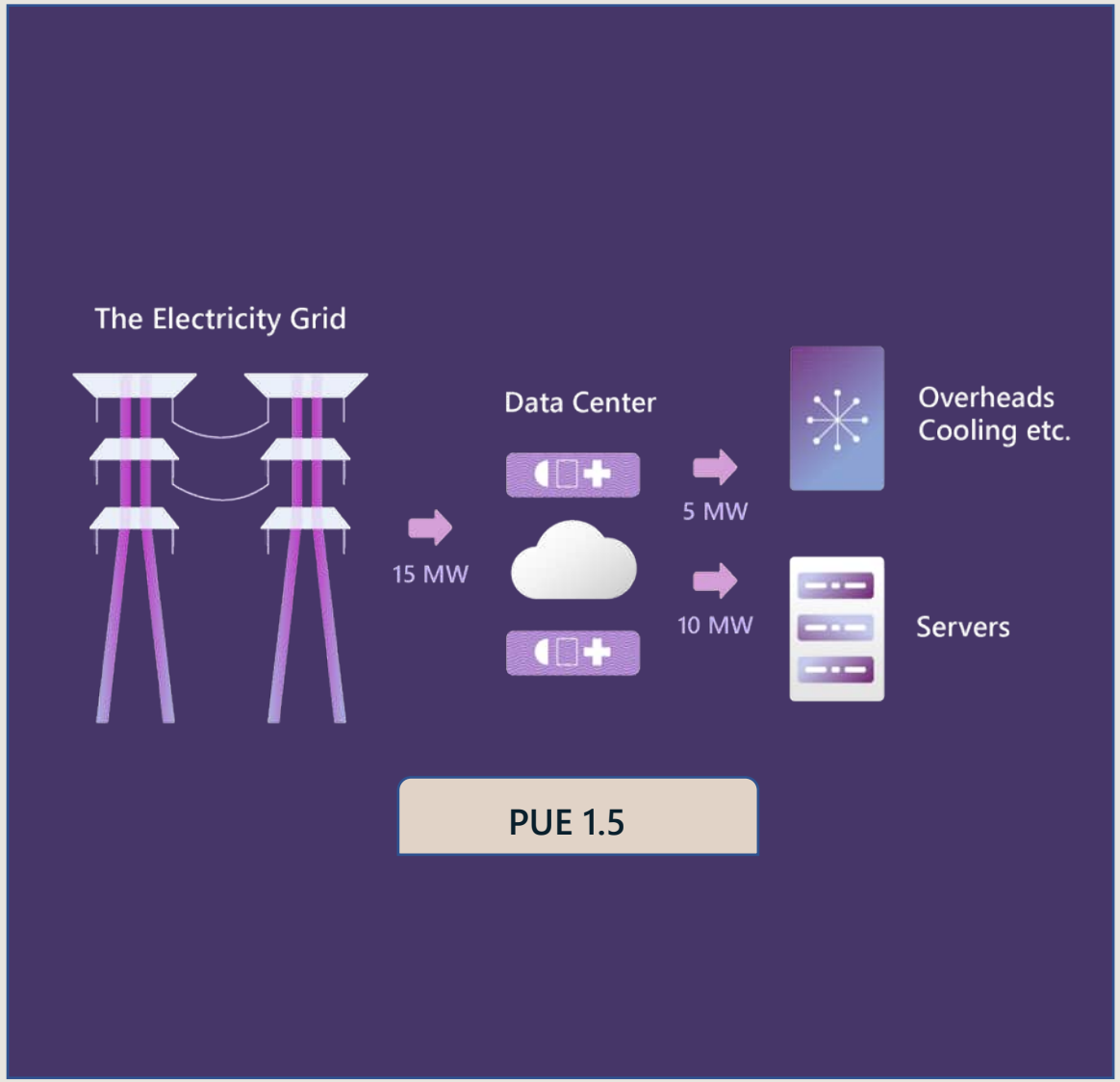


Principle 1:
Energy Efficiency

Power usage effectiveness

The power usage effectiveness (PUE) metric is widely used by the datacenter (DC) industry to measure energy efficiency. Specifically, PUE is determined by dividing the total amount of power entering a DC by the power used to run the IT equipment within it. When a DC's PUE is closer to the theoretical ideal case of 1.0, the energy efficiency of the DC infrastructure is deemed better as it indicates a higher proportion of the total available power is used by the IT equipment, and a relatively smaller amount is used for cooling and other overheads during the operations.

For example, if the actual power from the grid into a DC is 15 MW and for a PUE of 1.5, it means 10 MW is used by the IT equipment running user applications in that DC, and the remaining 5 MW is used by the cooling and other infrastructure overheads.

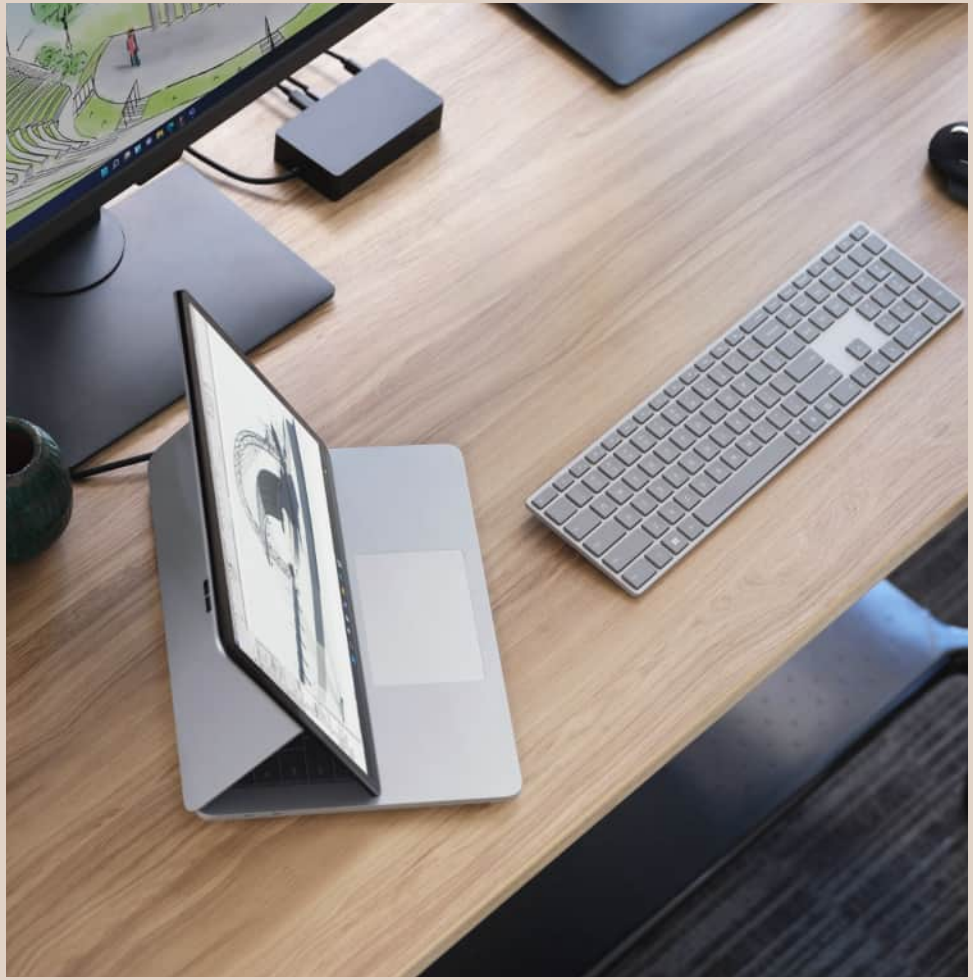


Feature 2:

Singapore standard for the deployment and operation of DC IT equipment

DCs may use up to 40% of its total energy consumption for cooling purposes. Traditionally, DCs are designed to operate at cooler temperatures to ensure optimal performance and reliability of IT equipment. The American Society of Heating Refrigerating, and Air-Conditioning Engineers provides guidelines for recommended temperature and humidity ranges for DCs, with the commonly accepted range being 18-27°C for IT equipment. With advancements in technology and improvements in DC designs, there has been growing interest in DCs operating at higher temperature points to reduce cooling overheads and improve energy efficiency. The latest generation of IT equipment can reliably operate at up to 35°C.

Studies have shown that 4% to 5% of cooling energy can be saved for every 1°C increase in IT equipment operating temperature. While this offers an opportunity for improvement, there is a lack of standard reference and best practices to help DCs in tropical countries to cool their IT equipment at higher temperatures. In light of this, IMDA has facilitated the development of Singapore Standard on the deployment and operation of DC IT equipment under a tropical climate. The standard is published as part of a Digital Connectivity Blueprint launched in June 2023. It offers an actionable methodology to DC operators keen on unlocking energy efficiency by operating at a higher temperature than current industry norms.



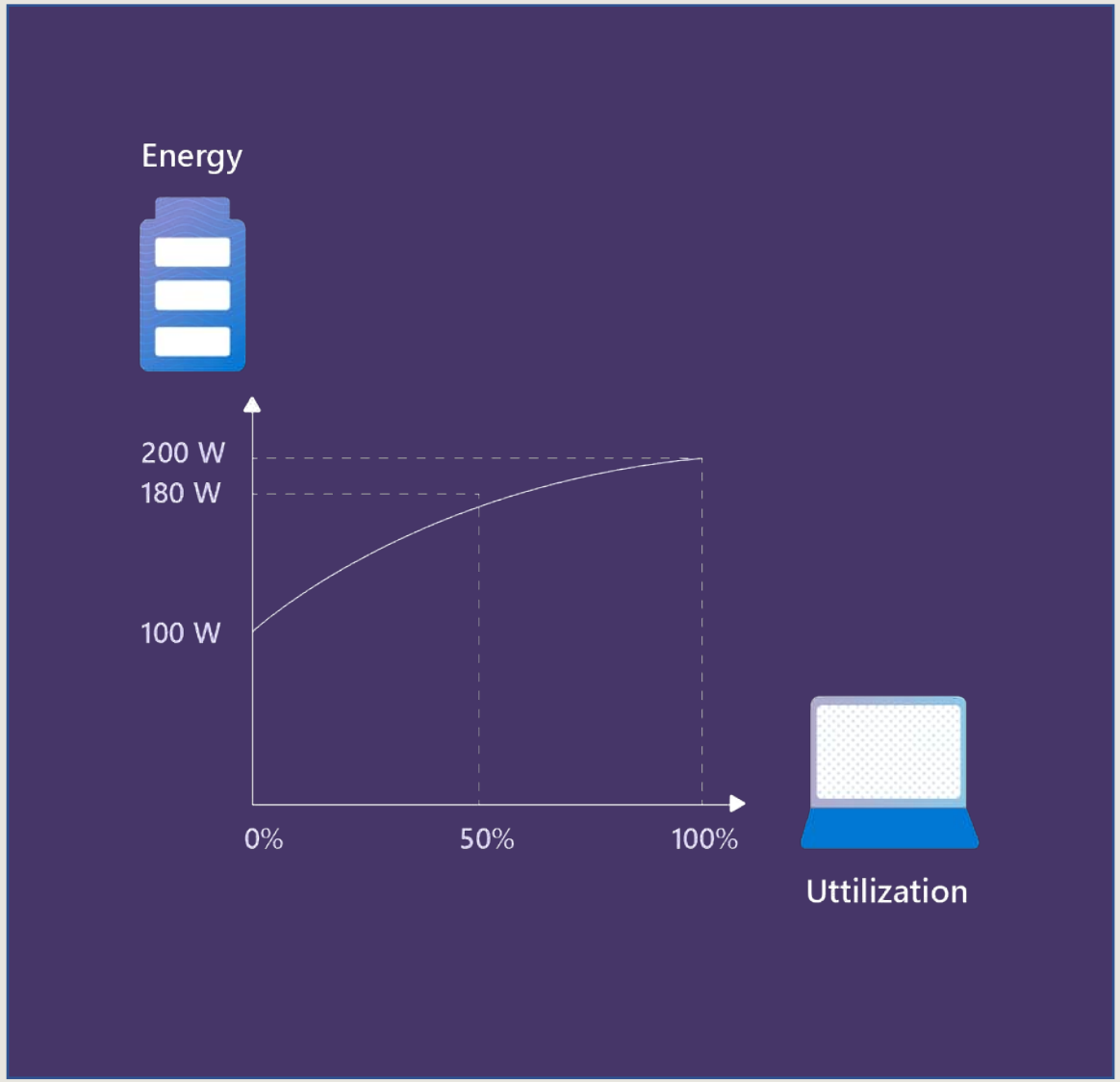
Principle 1:
Energy Efficiency

Energy proportionality

Energy proportionality measures the relationship between power consumed by a computer and the rate at which useful work is done (i.e. its utilisation). Utilisation measures how much of a computer's resources are used, usually expressed as a percentage. A fully utilised computer running at its maximum capacity has a high percentage, while an idle computer with little utilisation has a lower percentage.

The relationship between power consumption and utilisation is not linear. Mathematically, proportionality between two variables means their ratios are equivalent. For example, at 0% utilisation, a computer can draw 100W; at 50%, it draws 180W; and at 100%, it draws 200W.

Because of this, the more we utilise a computer, the more efficient it becomes at converting electricity to practical computing operations. One way to improve hardware efficiency is to run the workload on as few servers as possible, with the servers running at the highest utilisation rate, maximising energy efficiency.



Principle 1:
Energy Efficiency

Static power draw

The static power draw of a computer is how much electricity is drawn when it is in an idle state. Static power draw varies by configuration and hardware components, but all parts have some static power draw.

This is one of the reasons that end-user devices have power-saving modes. If the device is idle, it will eventually trigger a hibernation mode and put the disk and screen to sleep or even change the CPU's frequency. These power-saving modes save 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 conservative power saving. Many use cases running on servers demand total capacity as quickly as possible because the server needs to respond to rapidly changing demands, which leads to many servers in idle modes during low-demand periods. An idle server has a carbon cost from both the embedded carbon as well as its inefficient utilisation.

Principle 1:

Energy Efficiency

Summary

Energy is a proxy for the corresponding carbon that is emitted, so building an application that is energy efficient is usually equivalent to building an application that is carbon efficient. Sustainable software takes responsibility for its electricity consumption and is designed to consume as little as possible.

Quantifying the energy consumption of an application is a step in the right direction to start thinking about how an application can operate more efficiently. However, understanding your application's energy consumption is not all there is. The hardware your software is running on uses some of the electricity for operational overhead – for instance, power usage efficiency (or PUE) when your applications are in the cloud and running at data centres.

The concept of energy proportionality adds another layer of complexity since hardware becomes more efficient at turning energy into useful operations the more they are used. Understanding this provides better insight into how software behaves with respect to energy consumption in the real world.

Principle 2:
Carbon Awareness

Not all electricity is produced in the same way. In different locations and times, electricity is generated using a variety of sources with varying carbon emissions. Some sources, such as wind, solar, or hydroelectric, are clean, renewable sources that emit little carbon.

On the other hand, fossil fuel sources emit carbon at varying degrees to produce electricity. For example, both gas and coal emit more carbon than renewable sources, but gas-burning power plants emit less carbon than coal-burning power plants.

Carbon awareness is the concept of doing more when more energy comes from low carbon sources and doing less when more energy comes from high carbon sources.

Principle 2:
Carbon Awareness

How to be more carbon aware

It is first important to understand the concept of carbon intensity, which measures how much carbon equivalent (CO_2e) is emitted per kilowatt-hour (KWh) of electricity consumed. The standard unit of carbon intensity is gCO_2e/kWh, or grams of carbon per kilowatt hour.

If your computer is plugged directly into a wind farm, its electricity would have a carbon intensity of close to 0 gCO_2e/kWh since a wind farm emits little carbon to produce that electricity. However, most people plug into power grids supplied with electricity from various sources. Once on a grid, you are unable to control which sources supply the electricity you are using. Your carbon intensity will be a mix of all the current power sources in a grid, both the lower and the higher carbon sources.

Being carbon aware means responding to shifts in carbon intensity by increasing or decreasing your demand. If your work allows you to be flexible with when and where you run workloads, you can shift accordingly by consuming energy when the carbon intensity is lower and pausing production when it is higher. For example, one can train a machine learning model at a different time or in a region with much lower carbon intensity.

Principle 2:
Carbon Awareness

Demand shifting

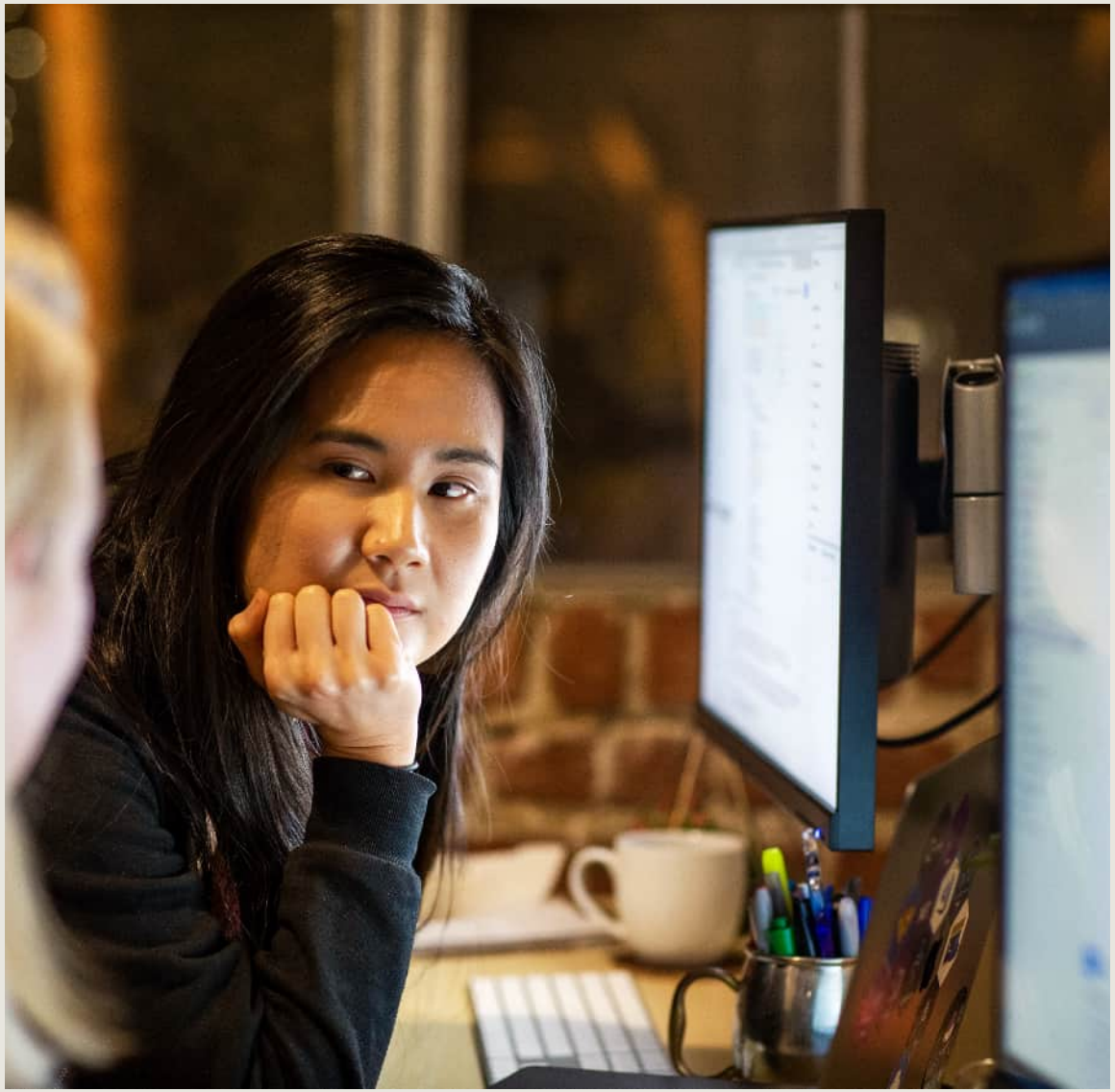
Demand shifting can be further broken down into spatial and temporal shifting.

Spatial shifting means moving your computation to another physical location where the current carbon intensity is lower. It might be a region that naturally has lower carbon sources of energy. For example, moving to different hemispheres depending on the season for more sunlight hours.

You can also adopt temporal shifting. For instance, if you can't shift your computation spatially to another region, another option you have is to shift to another time. Perhaps later in the day or night when it is sunnier or windier and, therefore, the carbon intensity is lower. This is complemented by advances in weather forecasting, which allows for better prediction of future carbon intensity.

Demand shaping

Demand shaping is a strategy of adapting energy consumption based on carbon intensity, in order to consume more in periods of low intensity and less in periods of high intensity. Being efficient with resources is one part of the equation, and we would also need to consume less at certain points to provide a holistic solution. A sustainable software solution may include an option to cancel a process when the carbon intensity is high, instead of merely demand shifting.



Feature 3:

Singapore GreenTech Challenge innovators to accelerate development and global adoption of innovative sustainability solutions.

Designed to accelerate the development and adoption of innovative sustainability solutions in Singapore, the Singapore GreenTech Challenge 2023 aims to help the nation progress towards its goal of achieving net-zero emissions by 2050.

The challenge, which reached over 3 million people and received 32 entries, unearthed new solutions for organizations to set and track sustainability targets, identify, and transition to renewable energy sources, and create, buy and sell tradeable carbon assets.

Setting and tracking sustainability targets – Evercomm
Evercomm assists businesses in managing energy as a strategic resource, tightly integrated with their production processes. Their world-first digital assurance platform, NXMap, serves as a carbon inventory and accounting tool, aligned with the ISO 14064-1 Standard and the GHG protocol to accurately calculate carbon emissions. With products such as NXOps and NXMap, Evercomm enables businesses to automate carbon footprint calculations down to the activity level. By integrating and standardizing various types of sustainability data, including dynamic and static environmental data, Evercomm empowers businesses to set and track progress towards their sustainability goals and guides them in their journey. Evercomm is an IMDA accredited company.

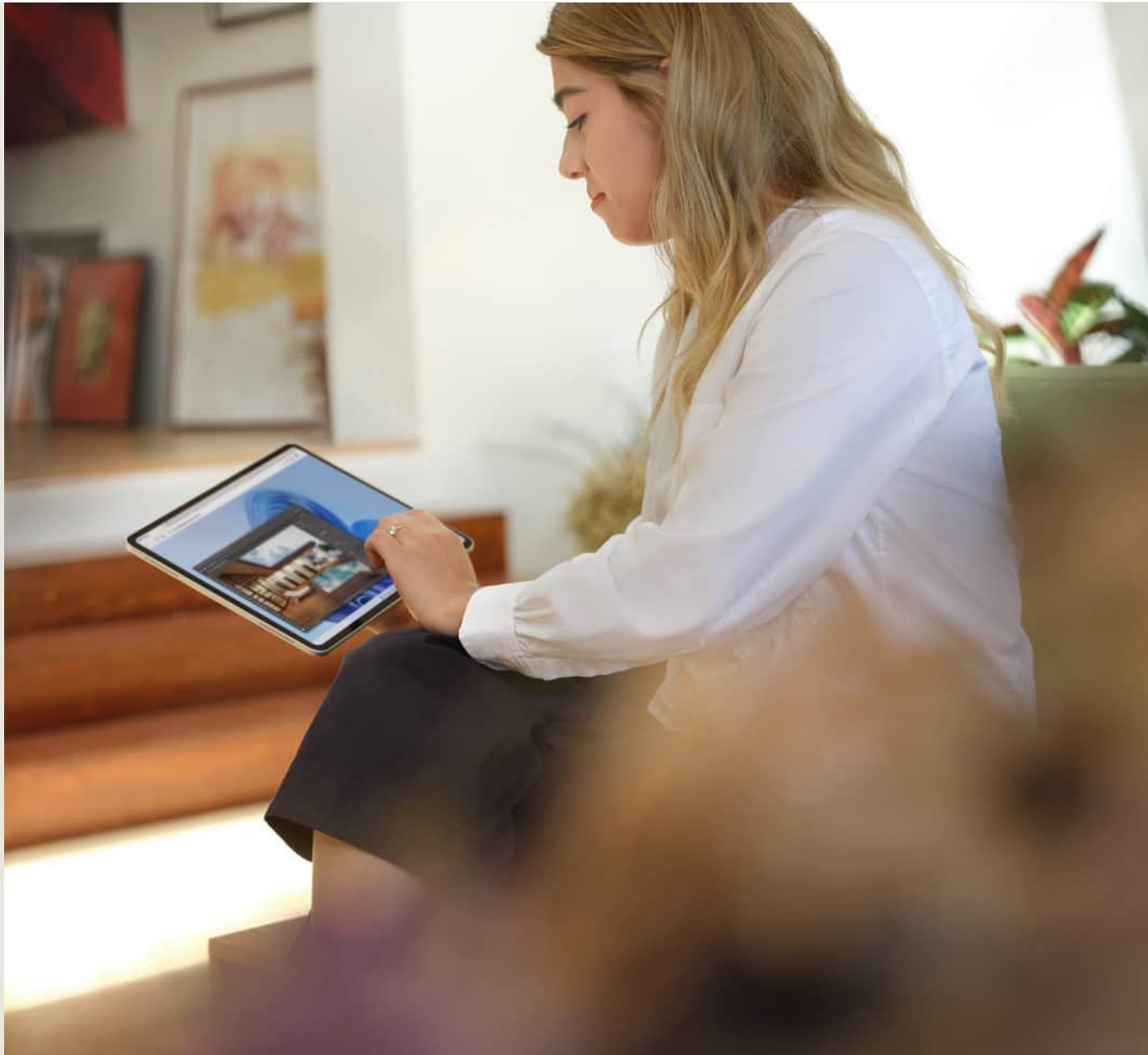
Identifying and transitioning to renewable energy sources – WeavAir
WeavAir offers real-time high accuracy emissions and offset analysis enabled by satellite and IoT networks. WeavAir's competitive advantage is in their unique real-time data source and collection. Their IP includes satellite imaging with proprietary access to satellite imagery from NASA. Analytics benchmarking indicators tracked by Global Reporting Initiative (GRI), UN SDGs, Singapore Exchange (SGX), Sustainability Linked Loans Principles (SLLP). Solutions that will help businesses identify renewable energy sources (e.g., solar, wind, hydro or low-carbon hydrogen) for potential use as they reduce carbon emissions.

Create, buy, and sell tradeable carbon assets – ReClimate
ReClimate's digital platform helps projects earn additional revenue from carbon credits and energy attribute certificates with minimal effort and upfront costs. With a focus on decarbonization in the EV and Renewable energy industry, ReClimate offers digital measurement, reporting, and verification, automating the matching of projects with financial institutions and investors to determine their eligibility for green financing options and other incentives.

Principle 3:
Hardware Efficiency

Embodied carbon is the amount of carbon pollution emitted during the creation and disposal of a device. When calculating the total carbon pollution for computers running software, both the carbon pollution associated with running the computer as well as the embodied carbon of the computer must be accounted for. For some devices, the carbon emitted during the manufacturing process may be higher than that emitted during usage. Any device, even one not consuming electricity, is responsible for the release of carbon over its lifetime.

If we take into account the embodied carbon and take hardware as a proxy for carbon emissions, being hardware efficient would go towards our goal of being carbon efficient. There are two main approaches to hardware efficiency. For end-user devices, we seek to extend the lifespan of the hardware. For cloud computing, we seek to increase the utilisation of the device.



Principle 3:
Hardware Efficiency

Extending hardware lifespan

A way to account for embodied carbon is to amortise the carbon over the expected life span of a device. For example, the building of a server resulted in the creation of 4000kg of CO_2e generated by the devices used to make the server, and we expect the server to then last four years. This means that the server emits 1000kg CO_2e/year on an amortised basis.

Hardware is retired when it breaks down or struggles to handle modern workloads. However, where possible, we can use software to build applications that run on older hardware and extend their lifespan.

Principle 3:
Hardware Efficiency

Extending hardware lifespan

A key way to increase hardware efficiency is by running systems at a high utilisation rate for processors, memory, disk space, and networking. It is more efficient to use a single server at 70% utilisation than 5 servers at 20% utilisation because of the cost of embodied carbon. In the same way that owning one car and using it every day of the week is much better than owning five and using a different one each day of the week, it is more efficient to use servers at their full capacity rather than employing several servers below capacity. Although emissions are the same, the embodied carbon that is used is much lower.

The most common reason for having under-utilised servers is to provide a buffer for peak capacity. However, spare capacity idling represents wasted embodied carbon. Being hardware efficient includes ensuring that devices are being utilised as much as possible for as long as possible.

One solution for this lies in the public hyperscale cloud. Due to the elasticity in capacity in the public cloud, there is ample capacity to cater for a need to scale up when needed. With multiple organisations making use of the public cloud and shared data centres, spare capacity can always be made available to whoever needs it, so that idle servers are minimised. The public cloud is thus generally far more efficient than a private on-premise solution.

03

Software Carbon Intensity Standard

It has often been said that you can't manage what you can't measure. The same is true of carbon emissions arising from software usage.

The Software Carbon Intensity (SCI) specification is a methodology developed by the Standards Working Group in the Green Software Foundation, designed to score a software application along a dimension of sustainability and to encourage action towards eliminating emissions.

It is a metric that helps software teams understand how their software behaves in terms of carbon emissions so they can make more informed decisions. It supplements the Greenhouse Gas Protocol, which is the most widely used greenhouse gas accounting standard internationally.

The Greenhouse Gas Protocol divides emissions into three scopes:

- **Scope 1**: Direct emissions from operations owned or controlled by the reporting organisation, such as on-site fuel combustion or fleet vehicles.
- **Scope 2:** Indirect emissions related to emission generation of purchased energy, such as heat and electricity.
- **Scope 3**: Other indirect emissions from all the other activities you are engaged in. Including all emissions from an organisation's supply chain; business travel for employees, and the electricity customers may consume when using your product.

While the GHG protocol calculates total emissions, the SCI calculates the rate of emission. Instead of bucketing the carbon emissions of software into scopes 1 to 3, the SCI buckets emissions into operational emissions (i.e. emissions from the running of software) and embodied emissions (i.e. emissions from physical resources required to run the software).

The SCI does not take into account purchasing of offsets in the form of neutralisations, compensations, or by offsetting electricity in the form of renewable energy credits. This means that an organisation that makes no efforts towards reducing their software-based emissions but simply purchases carbon credits cannot achieve a good SCI score.

If you make your application more energy efficient, hardware efficient, or carbon aware, your SCI score will decrease. Adopting the SCI as a metric for software development along with adherence to the GHG protocol, will aid in measuring the overall effectiveness of a sustainable software solution.

How to calculate the SCI score

1 Decide what to include

What software components to include or exclude in the SCI score means defining the boundaries of your software; where it starts and where it ends. For every software component you include, you will need to measure its impact. For every major component you exclude, you need to explain why.

The SCI specification doesn't currently prescribe what to include or exclude. However, one should include all supporting infrastructure and systems that significantly contribute to the software operation. The software boundary should be disclosed when the SCI score is reported, as the score is influenced by this definition.

2 Choose your functional unit

As discussed, the SCI is a rate, rather than a total. It measures the intensity of emissions according to the selected functional unit.

The SCI specification doesn't prescribe the functional unit and you are free to pick whichever best aligns with how your application scales. For example, if your application scales by the number of users, this could be a viable functional unit.

Future iterations of the SCI might suggest specific functional units for different types of applications to aid with standardisation. For instance, it may make sense for streaming applications to use time (e.g. minutes) as their functional unit.

The SCI equation:

$$SCI = ((E*I) + M) \text{ per } R$$

E = Energy consumed by a software system
I = Location-based marginal carbon emissions
M = Embodied emissions of a software system
R = Functional unit (e.g. per additional user, per API-call, per ML job, per minute)

The equation can be further simplified to:
SCI = C per R (Carbon per R)

How to calculate the SCI score

3 Decide how to measure your emissions

The next step is to decide how you will quantify the emissions of each software component. There are two methods of quantification: measurement and calculation.

Measurement is using counters of some form. For example, measuring the energy consumption of your software component by using a hardware device in the wall socket. Or using counters on hardware that directly measure energy consumption. If you can directly count your units, you should use the measurement approach.

Calculation involves indirect counting, often using a model. For instance, if you cannot directly measure your application's energy consumption, but instead have a model that estimates the energy consumption based on the CPU utilisation, this would be considered a calculation approach.

4 Calculate the SCI score

Using the methodology described in the previous steps, start to calculate the SCI score for each software component in your boundary. The total SCI score for the software application is the combined score of all the different components.

Multiple SCI scores for the same application can be calculated. The SCI score is helpful to understand how an application behaves with respect to carbon emissions in different scenarios. For example, a streaming application might choose carbon per minute as a first metric. It might also calculate the carbon per user per day. The carbon per $ revenue metric might provide yet another helpful dimension.

The SCI equation:

$$SCI = ((E*I) + M) \text{ per } R$$

E = Energy consumed by a software system
I = Location-based marginal carbon emissions
M = Embodied emissions of a software system
R = Functional unit (e.g. per additional user, per API-call, per ML job, per minute)

The equation can be further simplified to:
SCI = C per R (Carbon per R)

04

Toolkit & Resources

There are several tools and resources available for developers to leverage when creating sustainable software solutions. Some of these are listed in this section, although they are non-exhaustive.

The Carbon Aware Software Development Kit (SDK)

Carbon awareness is one of the key principles of sustainable software development. The Carbon Aware Software Development Kit is a web API and command line interface released by the Green Software Foundation and the Foundation's Open Source Working Group, which helps in building carbon-aware software solutions with the intelligence to use the greenest energy sources. This allows for demand shifting, including running at the greenest time and/or location, and captures consistent telemetry and reports on emissions reductions to allow for informed decisions to be made.

The Carbon Aware SDK is available on GitHub.

Sustainable tools in the cloud

As earlier discussed, the use of the hyperscale public cloud is an efficient way to run workloads in line with sustainable software principles. There are several best practices and core competencies that one can adopt when utilising cloud solutions, some of which are outlined in the various resources below:

- Several relevant articles have been penned by the Green Software Foundation, including:
  - A five-part series on sustainable tech choices for cloud:
  - A column on sustainable systems, including hardware efficiency practices.
  - 10 recommendations for green software development.
- Developers can take reference from several patterns from the Green Software Patterns Catalogue, which is an online open-source database of sustainable software patterns categorized.
- Microsoft Azure's Well-Architected Guidelines allows a user to improve the quality of their workloads through adherence to five architectural pillars, including reliability, security, cost optimisation, operational excellence and performance efficiency. The guidelines includes design principles, application design and platform considerations for sustainable workloads.
- Microsoft Sustainability Manager is an extensible solution that unifies data intelligence and provides comprehensive, integrated, and automated sustainability management for organisations at any stage of their sustainability journey. It automates manual processes, enabling organisations to more efficiently record, report, and reduce their emissions.
- Targeted reporting tools, such as the Microsoft Emissions Impact Dashboard, which estimates carbon emissions arising from Microsoft cloud services.
- Sustainable software engineering practices in Azure Kubernetes Service.
- Microsoft developer blog on sustainable software.
- Azure's cost analysis tool allows for right-sizing and optimisation of resources.
- Azure Machine Learning resource metrics, which help customers understand the computational and energy costs of their artificial intelligence workloads.

Learning pathways

The Linux Foundation provides an initial certification course for green software practitioners.

For an introduction to the principles of sustainable software engineering, peruse the following learning pathway curated by Microsoft Learn.

Feature 4:

Architecting Singapore's green digital future together

The Singapore's Green Plan 2030 is a whole-of-nation movement aimed at advancing a sustainable future for the city-state, fuelled by innovation, technology, and open collaboration.

In June 2023, Singapore's Infocomm Media Development Authority (IMDA), announced its interest to foster collaboration to design, build and operate lower carbon footprint digital solutions for a sustainable digital future. IMDA advocates for green software and invests in building up digital capabilities to measure and implement low-carbon digital solutions. Singapore-based tech providers are encouraged to switch to using greener techniques that drives optimisation in the usage of their solutions.

Singapore is the first country to join the Green Software Foundation through the IMDA.

05

Recommendations for Developers

The creation of sustainable software solutions is a journey that starts at an early stage of software and product development, and involves an intentional consideration of the relevant environmental, social and economic impact of software, while taking steps to reduce its potential negative effects.

Developers may consider the principles, tools and resources set out in this guidelines paper to drive their sustainability journey. A list of recommendations and best practices are listed below:

- **Leverage tools to measure software carbon emissions:** Use the Software Carbon Intensity standard and other reporting tools in calculating the carbon emissions of your software solution and optimising outcomes.
- **Minimise resource consumption:** Design software that uses minimal CPU, memory, and storage resources.
- **Use renewable energy:** Use renewable energy sources to power the infrastructure that hosts your software or select such sources if you are using a third-party vendor.
- **Reduce embodied carbon:** Minimise the environmental impact of underlying processes linked to the software solution, such as using low-carbon transportation and choosing eco-friendly supplies.
- **Promote energy efficiency:** Design software that helps users conserve energy, such as by enabling power-saving modes, providing information about energy-efficient settings or engaging in demand shifting and shaping.
- **Educate users:** Provide information to users about how they can use your software in an environmentally responsible way, including various eco-friendly options.
- **Collaborate with others:** Work with other developers, companies, and organisations to promote the creation of sustainable software and to reduce the environmental impact of the technology industry.

There are further sustainable software actions that can be applied to the development and deployment of software:

- **Use open-source software:** Open-source software can often be more energy-efficient and environmentally friendly than proprietary software, as it allows for community-driven development and collaboration, rather than being dependent on a single provider.
- **Choose cloud-based solutions:** Cloud-based solutions are likely to be more energy-efficient than on-premises software, as they allow for the sharing of resources and infrastructure among multiple users.
- **Optimise for minimal hardware requirements:** Design software to run efficiently on low-power devices and with minimal hardware resources, to reduce energy consumption and material use.
- **Use virtualisation:** Virtualisation can allow multiple applications and operating systems to run on a single physical machine, reducing the number of hardware resources needed.
- **Consider the use of renewable energy:** Use renewable energy sources to power the development, testing, and deployment of software, or offset the carbon emissions of non-renewable energy sources.

Developers who adopt the principles and best practices above and demonstrate a positive sustainability outcome may in due course benefit from certain funding support and grants from the IMDA, subject to applicable conditions.

06

Appendix

Glossary

CO_2e – Carbon dioxide equivalent
ICT – Information and Communication Technology
IMDA – Infocomm Media Development Authority of Singapore
KWh – Kilowatt-hour
PUE – Power Usage Efficiency
MW – Megawatt
SCI – Software Carbon Intensity
SDK – Software Development Kit

Acknowledgements

We would like to thank the following individuals and organisations who provided valuable input for the development of this Sustainable Software Development Guidelines:

- Green Software Foundation
- BizTech Group, Infocomm Media Development Authority
- Tammy McClellan, Senior Cloud Solution Architect, Microsoft
- Sakthivel Nachimuthu, Senior Cloud Solution Architect, Microsoft Singapore
- Yik Wai Yi, Communications Lead, Microsoft Singapore
- Jeth Lee, Chief Legal Officer, Microsoft Singapore

References

- https://news.microsoft.com/en-sg/2023/06/28/singapore-greentech-challenge-2023-winners-set-to-accelerate-development-and-global-adoption-of-innovative-sustainability-solutions/
- https://www.evercomm.com.sg/home
- https://weavair.com/
- https://reclimate.earth/
- https://www.imda.gov.sg/Content-and-News/Press-Releases-and-Speeches/Press-Releases/2022/Singapore-Advances-Plans-to-Become-a-Leading-Innovation-Digital-Sustainability-Hub-in-Southeast-Asia
- https://news.microsoft.com/en-sg/2022/11/17/singapore-advances-plans-to-become-a-leading-innovation-digital-sustainability-hub-in-southeast-asia/
- https://news.microsoft.com/en-sg/2023/06/28/singapore-greentech-challenge-2023-winners-set-to-accelerate-development-and-global-adoption-of-innovative-sustainability-solutions/
- https://patterns.greensoftware.foundation/
- https://www.energystar.gov/products/raise_temperature
- https://grnsft.org/sci
- https://github.com/Green-Software-Foundation/carbon-aware-sdk
- https://greensoftware.foundation/articles/sustainable-tech-choices-for-cloud
- https://greensoftware.foundation/articles/sustainable-systems-mastering-the-tradeoff-between-high-server-utilization-and-ha
- https://greensoftware.foundation/articles/10-recommendations-for-green-software-development
- https://patterns.greensoftware.foundation/
- https://learn.microsoft.com/en-us/azure/architecture/framework/
- https://learn.microsoft.com/en-us/industry/sustainability/sustainability-manager-overview
- https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard
- https://learn.microsoft.com/en-us/azure/aks/concepts-sustainable-software-engineering
- https://devblogs.microsoft.com/sustainable-software/
- https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/quick-acm-cost-analysis
- https://techcommunity.microsoft.com/t5/green-tech-blog/charting-the-path-towards-sustainable-ai-with-azure-machine/ba-p/2866923
- https://training.linuxfoundation.org/training/green-software-for-practitioners-lfc131/
- https://learn.microsoft.com/en-gb/training/modules/sustainable-software-engineering-overview/