Is Sustainable Software a Distraction or an Imperative?

Published 30 March 2022 - ID G00766182 - 17 min read

By Analyst(s): Nick Jones

Initiatives: Executive Leadership: Sustainability; Technology Innovation

Sustainable software is a business decision, not an IT decision. Executive leaders must plan sustainable software investments in the context of their organization's overall sustainability goals and prioritize those that have a material impact.

Overview

Key Findings

- Software sustainability is a business decision, not an IT decision. Through 2025, it will primarily be concerned with increasing energy and resource efficiency as well as reducing carbon emissions.
- Many tactics to make software more sustainable exist, but they often involve business trade-offs that are difficult to quantify because of lack of information or architectural complexity.

Recommendations

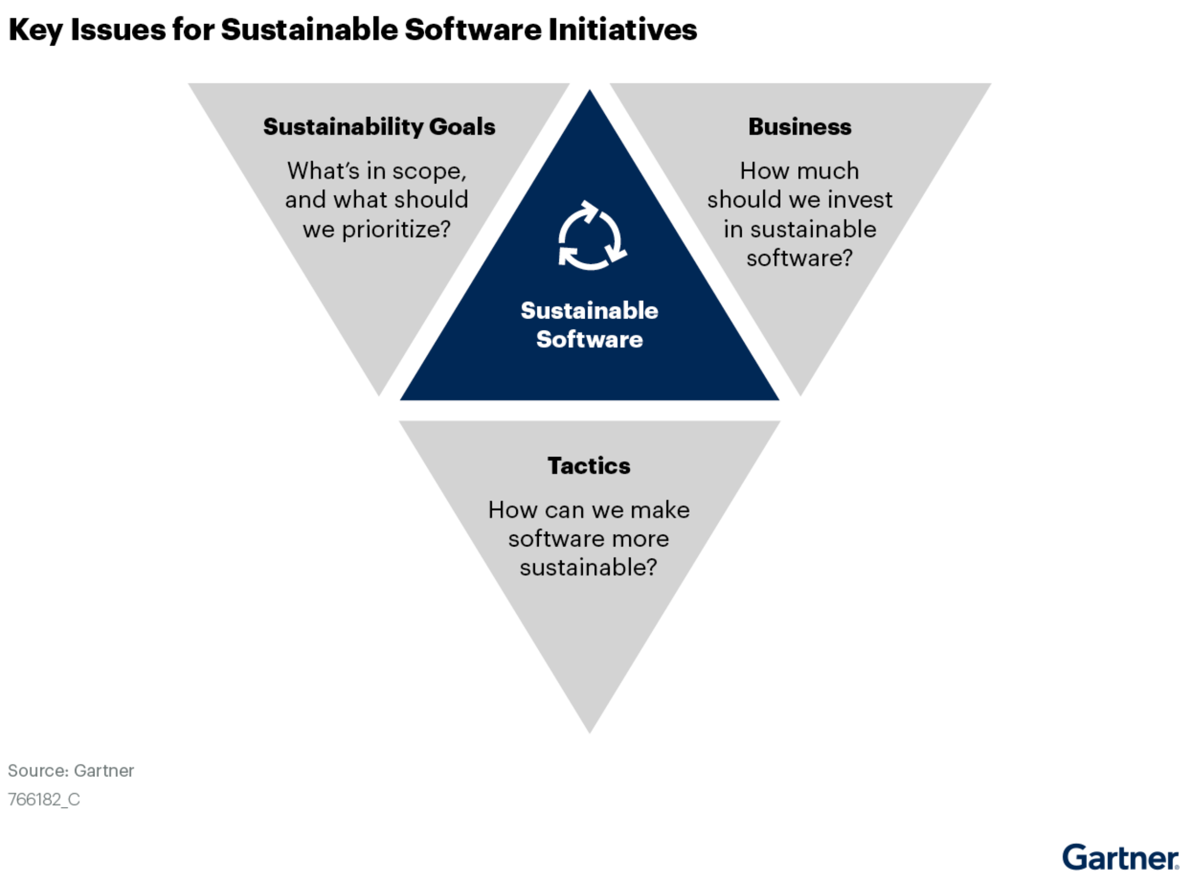
Executive leaders in charge of sustainability should:

- Develop a sustainable software strategy in collaboration with business stakeholders by identifying where software sustainability matters to them, what trade-offs they'll accept to achieve it and what success metrics they require.
- Identify quick wins and pragmatic tactical benefits by prioritizing effort on "big ticket" systems that will have a material impact on the organization's sustainability targets.

Introduction

In 2021, an ACM technology brief estimated that the information and communication technology (ICT) sector contributed between 1.8% and 3.9% of global carbon emissions. [1] As organizations aim to improve their sustainability credentials, they will inevitably consider the impact of hardware and software. This raises the question of what "sustainable software" or "green software" is and how to improve software sustainability. Any organization considering the topic must understand the three basic issues illustrated in Figure 1.

Figure 1. Key Issues for Sustainable Software Initiatives



The three key issues are:

- **How do we define "sustainable software"?** Software sustainability is a complex concept that focuses on energy and resource efficiency. For some organizations, it extends to social, environmental and governance issues, which are far outside the IT organization's responsibility. Choosing an achievable definition of "sustainable software" for the enterprise and identifying goals, metrics and system boundaries are critical.

- **How do we select the right tactics and technologies?** Many ways to make software more sustainable exist, and some don't involve changing the software (e.g., powering hardware with zero carbon electricity).

- **What's the most effective investment?** Software sustainability is a business investment that must be planned in the context of the organization's wider sustainability initiative. The organization must balance sustainability investments in ICT with those in other areas of its operations to achieve the best overall outcome.

The impact of ICT on the organization's overall sustainability footprint depends on its other business activities. If greenhouse gas (GHG) emissions from industrial processes vastly exceed those from ICT, green software may not be a sustainability priority. However, organizations using compute-intensive software for simulation or AI training might identify opportunities for software or hardware optimization to have a material impact on their overall carbon footprint

Sustainable software is a business decision, balancing the needs of many stakeholders such as customers, investors, staff, regulators or partners. Many software sustainability investments imply business trade-offs; for example, making a website more energy-efficient might involve curtailing the use of video, which could impact online sales.

Software sustainability will also be a moving target. In 2022, most definitions focus on Scope 2 issues — energy and carbon footprint, hardware sustainability, and in a few cases, recycling e-waste. In the longer term, Scope 3 emissions are likely to become more important (see Note 1).

Finally, in this research, we're considering only the sustainability footprint of running software, not developing it. We're also not addressing the wider opportunities to use software for improving the sustainability of other corporate activities and processes.

Analysis

Sustainability Is a Business Decision

Software Sustainability Is Derived From Materiality

Any definition of sustainable software must be derived from the organization's overall sustainability goals. A key concept here is materiality; i.e., what actually matters to the organization. This requires that you consider sustainability from the perspective of two axes: business impact and stakeholder impact. Stakeholders might include customers, government, investors, activists, staff and so on.

The goal is to focus effort on areas that are most important to stakeholders and where there is a large business impact. This is especially important in the context of software sustainability, where many potential actions may be well-meaning but not material. See Quick Answer: 3 Reasons to Be Cautious About Investing in Environmentally Sustainable Software for more.

Ask the Key Business Questions

Software sustainability is ultimately a business issue involving business trade-offs, benefits and costs. For example, even simple tactics such as scheduling intensive processing for times when more sustainable energy is available could have business implications. Therefore, it's essential to define goals and trade-offs with business stakeholders before choosing tactics. Key questions include:

- How significant is ICT in terms of our overall sustainability footprint, and what must we change to make a meaningful impact on the organization's targets? In the short term, the focus is likely to be on energy consumption, carbon emissions and e-waste. In the longer term, the definition may broaden.
- Who are the stakeholders, and what do they want? How and where do ICT emissions impact those stakeholders?
- Whose sustainability are we trying to improve? This might include (but isn't limited to) business units, home workers or customers. Technical tactics may differ depending on stakeholder needs.
- What are the business costs and trade-offs? These could include the commercial impact such as of modifying websites, operational costs such as switching to greener power or opportunity costs if money is to be spent reengineering existing programs or processes.
- How will we measure success? What metrics are we trying to achieve, and how will they be measured?
- How will we explain our decisions and actions? A lot of PR is associated with sustainability, and the trade-offs and investments we make must be explained to stakeholders.

The key point here is that investments in sustainable software are just that — investments. As such, they don't stand alone but will be one of a larger set of candidate investments to improve overall sustainability.

Justify Investments in More-Sustainable Software

In 2022, approaches to select and justify sustainable software investments are very immature and incomplete. Some approaches include:

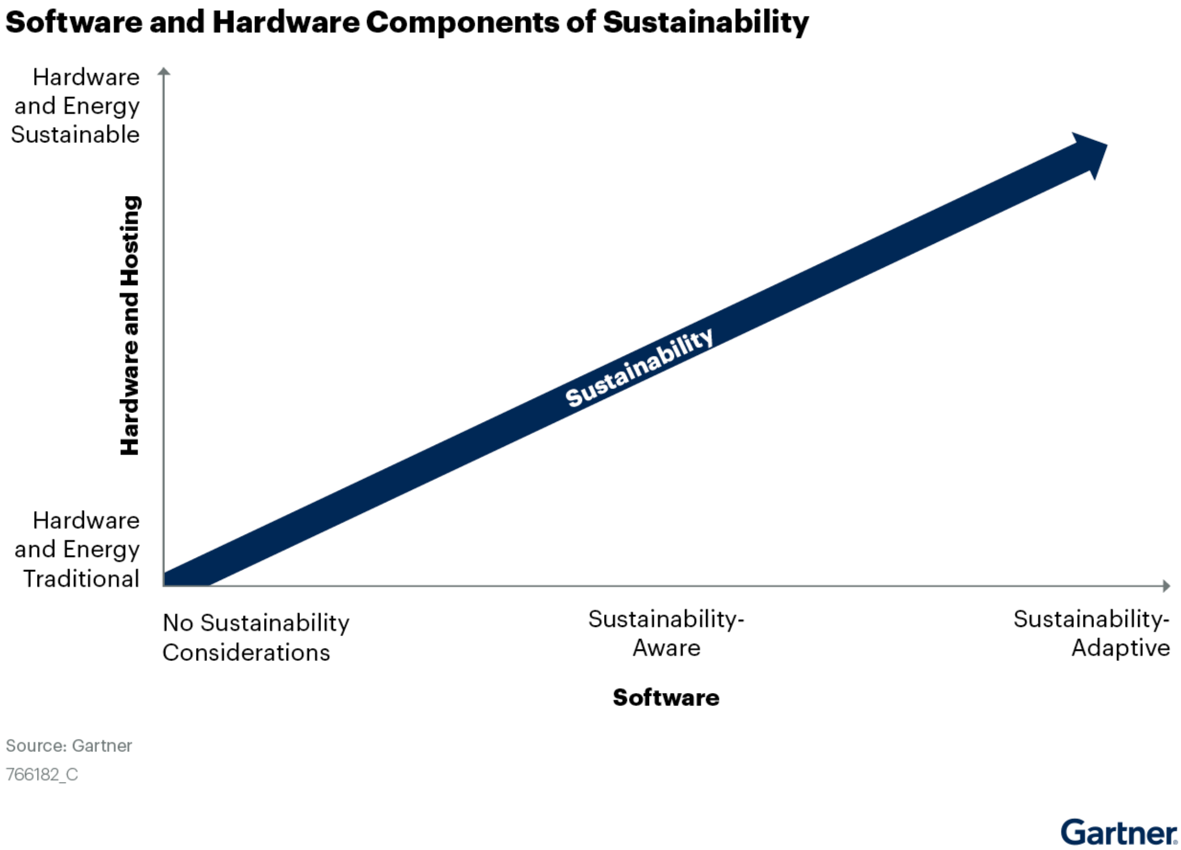
- **Fix obvious “big ticket” items.** Some organizations may own software or systems that are obvious large-scale GHG emissions culprits. These might include AI training, simulation or blockchain mining. The business case may involve the cost of energy they use, or it is taken as an overhead because the business doesn’t want the brand impact of being seen as environmentally unsound.
- **A need to meet external targets.** These could be defined by regulators, government, customers or even internal sources. Such cases may be non-negotiable, so don’t involve classic financial justifications.
- **Shadow carbon pricing.** Some organizations set a shadow carbon price, which acts as a hypothetical surcharge on projects and is used as a tool to compare investment alternatives.
- **Internal carbon price.** Some organizations set a price on carbon-generating activities, which acts as an internal carbon tax. Such taxes may not reflect the true external market price of carbon emissions but do serve as levers to control some business behavior.
- **Act of good faith.** Some ICT sustainability is basically an act of faith because it’s seen as the right thing to do, especially if the cost is relatively low (e.g., architecting software for improved sustainability when developing new systems).

Focus Your Effort Where It Will Make a Difference

Software and Hardware Must Be Considered Together

From a sustainability perspective, software and hardware are inseparable (see Figure 2). Sometimes the path to improved ICT sustainability will be via more energy-efficient hardware or more-sustainable power sources for existing hardware. In other cases, software changes may reduce emissions or enable extending the life span of older hardware, deferring the sustainability impact of new equipment purchases. New hardware will typically deliver significantly more processing power per watt of energy but comes with an embodied carbon and material cost.

Figure 2. Software and Hardware Components of Sustainability



From the hardware perspective, the journey is one toward:

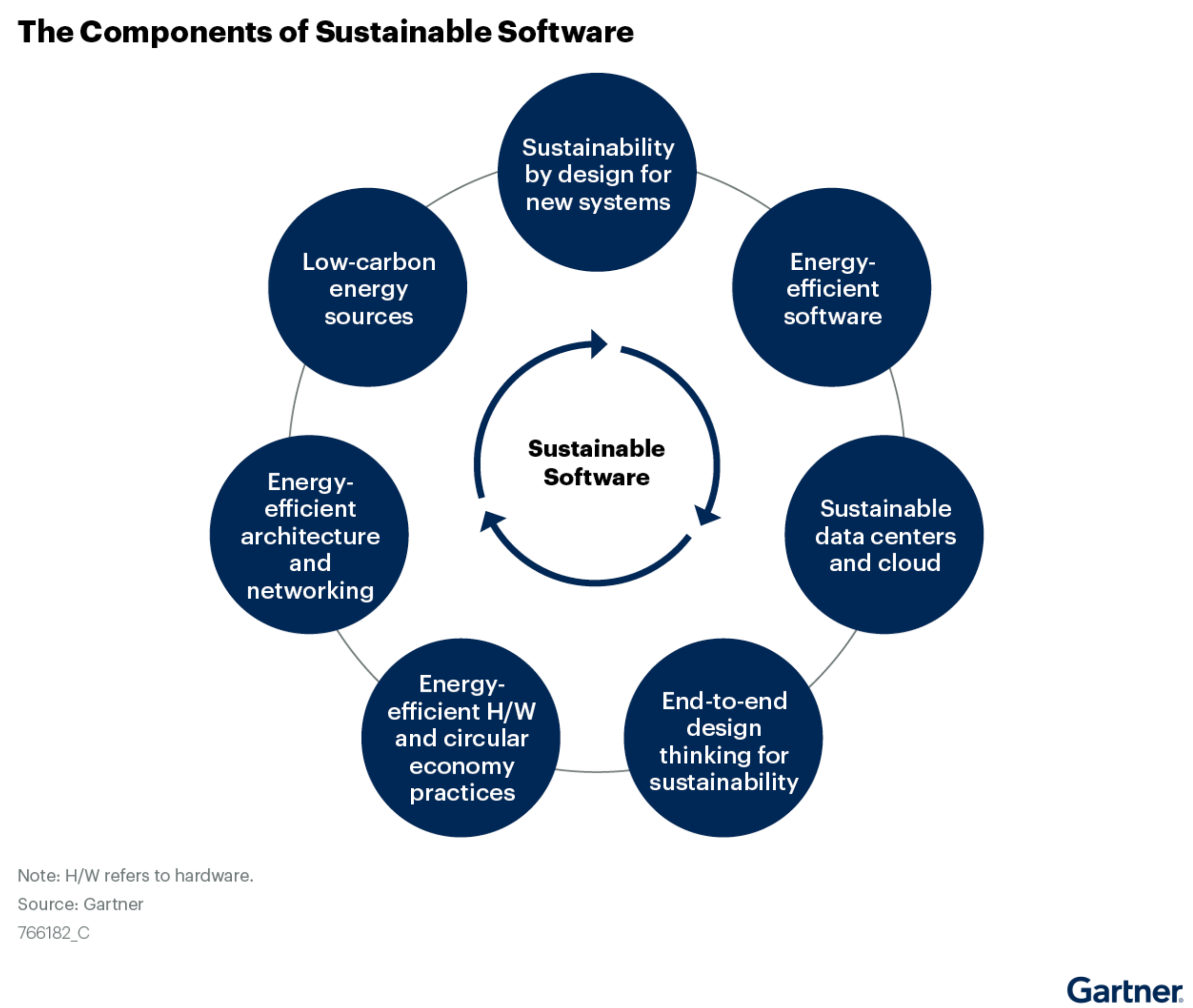
- Improved operational energy efficiency
- Hardware powered by sustainable energy such as renewables
- An ecosystem that extends the life span of devices through upgradability, maintainability, component reuse and circularity, avoiding premature obsolescence
- Reduced full life cycle carbon footprint
- Superior and ethically sourced materials

From the software perspective, the journey has three key points. The first is software, which is not designed with sustainability in mind. The second is software engineered for sustainability. The third is software that is aware of its sustainability context and adapts accordingly. For example, future platforms may provide APIs for software to determine the source of its energy or participate in a demand management ecosystem by adapting its features dynamically.

Sustainable Software Strategy

The main components of a sustainable software strategy are illustrated in Figure 3.

Figure 3. The Components of Sustainable Software



In this research, we don’t intend to discuss these practices and technologies in detail, but we will illustrate some opportunities and issues. A sustainable software strategy will involve education, technology, tools and processes to support all of these areas. In the sections below, we outline some of the key issues in four of them.

Sustainable Programming

Many programming tactics and tools can reduce the energy used by software, although as with any form of optimization, only a few sections of code will likely warrant serious effort. Appropriate tactics depend on the platform and application architecture, but a few examples include:

- **Algorithms and data structures** — Different algorithms for the same task may have very different efficiencies.
- **Programming languages** — Studies have shown that some programming languages use 70 times more energy for the same task than others. [2]
- **Application architecture** — For example, effective use of multithreading can make programs more energy-efficient.
- **Programming patterns** — Certain programming patterns and tactics are much more inefficient than others; for example, tactics that defeat the operating systems energy efficiency measures.

The maturity of energy-efficient programming varies widely. In domains such as the Internet of Things (IoT) and mobile applications, programmers have been aggressively optimizing energy consumption for decades. However, these domains are small consumers of energy and not a priority from a corporate sustainability perspective. Cloud providers can often provide information on energy consumption and how to optimize it. The energy consumption of private data centers is somewhat understood, if only because the organization pays the electricity bill.

But in other areas such as PC programming, the practices are less mature. For example, the tools to provide data on a program's energy consumption tend to be weak. Understanding the power consumed by processing is relatively easy, but extending that view to disks, displays, peripherals and networking is challenging.

The examples also illustrate the trade-offs that IT leaders must make. Some of the languages that are the most inefficient in terms of energy consumption are also among the most popular and productive from a programming perspective, such as Python or Ruby.

We expect there will be few cases where rewriting existing code to improve its sustainability will be justified because of the cost and risk of redevelopment and retesting. But sustainable programming tools and practices are likely to become part of IT's standard toolkit for new development work.

The Role of Hardware

Historically, computer performance per watt of power consumed has improved continually as smaller silicon geometries are deployed and chip design is improved. In the past, the power efficiency of computers has doubled (very approximately) every three years. So one of the most effective ways to make the same program consume less energy is to run it on more modern hardware.

However, this incurs the embodied energy cost of hardware replacement. Organizations must make carefully balanced decisions about hardware life cycles, focusing on early replacement of energy-intensive devices such as servers, some of which generate over half their lifetime carbon emissions in the in-use phase.

Over the next few years, hardware efficiency improvements will continue following hardware improvements and research in areas such as energy proportional computing (see Note 2). These might provide approximate improvements of two to five times in energy consumption every few years, depending on the hardware being replaced.

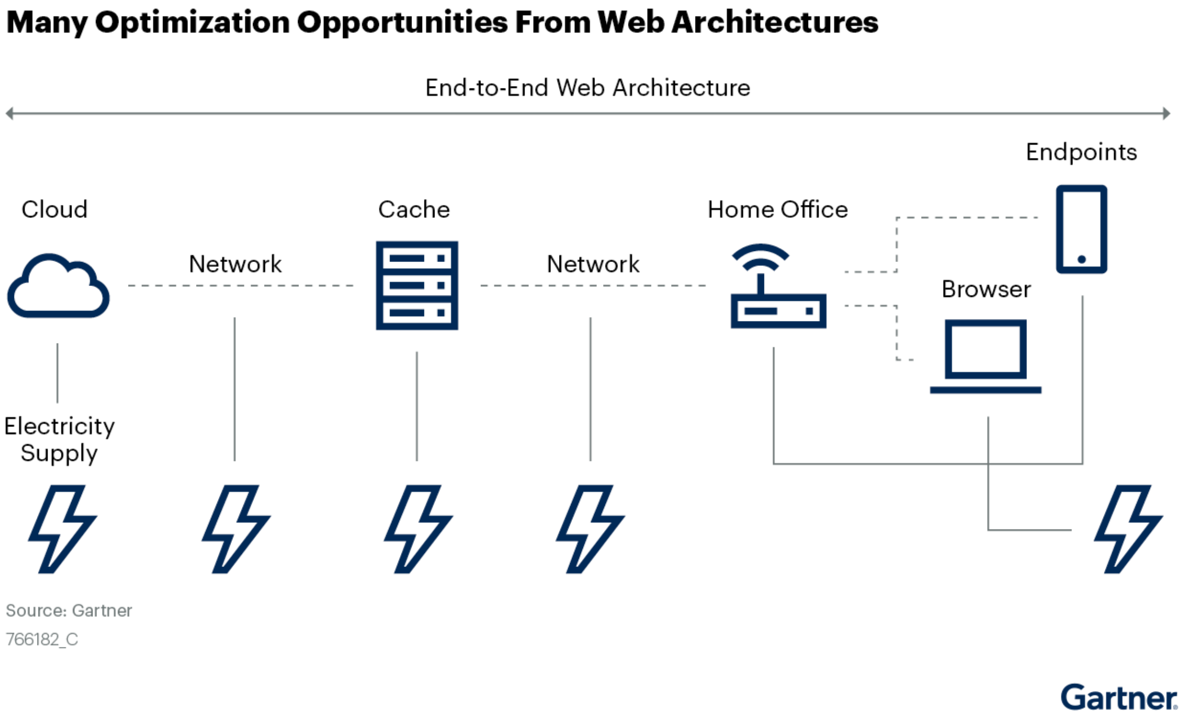
In a few situations, much more dramatic energy saving may be possible by using hardware optimized for specific tasks. In the short term (a year or two), technologies such as field-programmable gate arrays (FPGAs), graphics processing units (GPUs) or silicon chips optimized for specific applications such as AI can demonstrate power efficiencies of two to 20 times better than a general-purpose processor performing the same task. New chip architectures using extreme parallelism involving hundreds of thousands of cores will provide supercomputer performance at power budgets less than the equivalent number of GPUs.

In the longer term — three to 10 years — optical computing systems and postsilicon technologies have the potential for even greater improvements (see Note 3).

Sustainable Websites

One area that has received a lot of attention is sustainable websites. The architecture of web systems is complex, as illustrated in Figure 4, but offers many points at which energy consumption can be optimized.

Figure 4. Many Optimization Opportunities From Web Architectures



Sustainable Web Opportunities

Websites can be made more sustainable by reducing the amount of computation and network traffic involved in serving, delivering and rendering the website on the client device. A few of the many techniques used here include:

- Optimizing image sizes and using more efficient image formats such as vector images instead of bitmaps
- Reducing or eliminating the use of video, especially autoplay video, or replacing video with animations, which are typically more power-efficient
- Simplifying page design, font choices and so forth to reduce transmitted data and the number of pages served
- Exploiting network caching effectively to reduce traffic
- Using low-carbon cloud hosting services

However, sustainable web design illustrates many of the complex technology and business trade-offs implicit in designing for sustainability, such as:

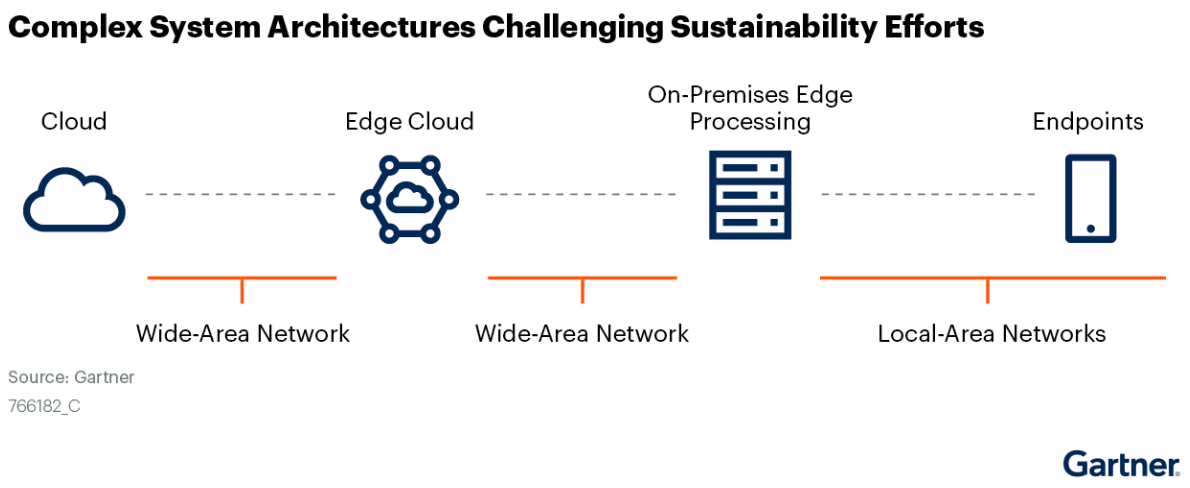
- Whose energy are we trying to save? Customer PCs, cloud services, telco network?
- Many energy-saving tactics impact the website design and user experience, and some may make it commercially less effective.
- Much of the end-to-end web architecture is opaque. For example, we don't know what network technologies are being used, what hardware the client owns, where caches are located or which elements use green power. The overall outcome is hard to quantify.
- Sustainable websites may deliver different user experiences and use different technologies to their predecessors. This demands staff and partners with new skills.

In the long term, we expect web optimization to become more important and more challenging because many of the technologies proposed for Web 3.0 and the metaverse have significant power consumption implications, such as 3D rendering of avatars.

Sustainability Demands End-to-End Architectural Thinking

Sustainable software design demands a design thinking perspective and an understanding of the end-to-end system architecture to identify where effort should best be expended. Unfortunately, this isn't always easy. As Figure 5 illustrates, ICT architectures can be very complex, and not all elements are visible to all the participants.

Figure 5. Complex System Architectures Challenging Sustainability Efforts



Modern architectures can have many components and stakeholders. A system such as the one shown above can involve cloud vendors, an edge cloud vendor such as a telco, on-premises processing such as gateways and a wide range of different computational endpoints varying from traditional PCs to IoT devices. All of these will be connected by multiple types of networks with different sustainability characteristics. Also, there will be a very diverse set of endpoints, some of which (such as IoT) will be outside our control and may pose new sustainability challenges, such as e-waste from batteries.

Optimizing such architectures from a sustainability perspective will involve complex trade-offs. For example, compressing data might reduce energy consumed by network traffic but increase that related to local processing. The location of processing required for latency and resiliency may conflict with the location that provides the best sustainability. Certain participants such as cloud services providers can often provide information about the carbon footprint of services, but some parts of these complex architectures may be opaque. In other words, you might not know what hardware and networks are being used and how to optimize the way you use them, except on very general principles such as "transmitting less data uses less energy."

In these situations, decisions about sustainability must be pragmatic, depending on what the organization can control and where there is sufficient information. Such decisions must be made by staff who have a broad understanding of the end-to-end architecture; for example, roles such as enterprise or application architects.

Conclusions and Challenges

Software sustainability will become an imperative for some organizations, especially those with compute-intensive workloads and/or those with ambitious GHG emission reduction targets. Through 2025, software sustainability will primarily be defined in terms of its energy consumption and carbon footprint, with many technical ways to reduce both. However, organizations should plan their sustainable software investments around the challenges and immaturity of the domain:

- Sustainable software is a business investment decision, not an IT decision. Plan your sustainability efforts in the context of the organization's wider sustainability targets and the impact of ICT on its overall sustainability footprint.
- Sustainable software often involves business trade-offs, such as in website design. Create a working group of IT and business stakeholders to define goals and success metrics.

- Apart from a few special cases, it will likely not be cost-effective to reengineer existing systems. Focus on making new systems more sustainable by providing tools, education and guidelines for developers.
- Do not consider software sustainability from only the perspective of software development because hardware and software are not separate decisions from a sustainability perspective. Many software sustainability improvements will come from better hardware.
- Put all the options on the table because there are many paths to the same goal. For example, make software more efficient, switch to more-efficient hardware, buy sustainable energy or reschedule processing to times when more-sustainable energy is available.
- Prioritize pragmatic achievable goals using the resources that are currently available. The task is difficult because information is scarce, and tools and resources are fragmented and incomplete. In 1H22, there are no authoritative green software engineering frameworks.
- Getting the data to make decisions may be difficult, especially in complex architectures involving services from partners. Seek improvements, not perfection.
- Create sustainability guidelines and reference architectures for key application domains. Few staff understand this issue at present, so train them.
- Look for quick wins because sustainability is a moving target.

Evidence

The information in this research has been obtained from discussions with peers and clients, academic papers, and Gartner surveys shown in the research listed in the Recommended Reading.

[1] Computing and Climate Change, ACM.

[2] Ranking Programming Languages by Energy Efficiency, Science of Computer Programming.

Note 1 - Scope Definitions

For reporting purposes, an organization’s greenhouse gas (GHG) emissions are classified into three categories or scopes:

- Scope 1 — Direct emissions. These relate to GHG emitted directly by the organization's facilities and equipment, such as vehicles or boilers and on-site manufacturing processes. These are seldom an issue from a software sustainability perspective.
- Scope 2 — Indirect emissions owned by the organization. These relate to GHG produced in the course of creating the energy purchased by the organization, such as electricity, fuel, heating and cooling.
- Scope 3 — Indirect emissions not owned by the organization. These relate to GHG associated with a wide range of upstream and downstream activities such as business travel, waste, capital goods, purchased goods and services.

Note 2 - Energy Proportional Computing

The term "energy proportional computing" refers to platforms whose energy consumption is proportional to the amount of computing work they are performing.

Note 3 - Nontraditional Computing Technologies Offer Sustainability Opportunities

Illustrations of energy savings through nontraditional computing now and in the future include:

- Special-purpose hardware architectures and platforms. Mainstream technologies in 2022 include, for example, Google's Tensor Processing Units (TPUs). Power efficiency varies depending on the application, but ranges of two to 20 times (compared to a general-purpose processor) have been demonstrated.
- New chip architectures involving extreme parallelism. Devices such as Cerebras cram 850,000 cores onto a single huge silicon chip. The manufacturers claim 100 times the performance of a GPU solution at one-third comparable energy consumption.
- Optical computing systems have demonstrated special-purpose algorithms (e.g., for AI) running at 10% of the energy cost of GPU solutions.
- In the long term (a decade), new postsilicon technologies are expected to enable devices with an order of magnitude better power consumption than silicon chips. Also, in this time frame, quantum computers will likely be able to perform computations that would be impossible with any form of traditional hardware, regardless of the amount of available energy.

Recommended by the Author

Some documents may not be available as part of your current Gartner subscription.

Quick Answer: How Can You Reduce the Environmental Impact of Data?

Define Sustainability and Leverage Materiality to Drive More Effective Strategy

Apply Digital Business to Sustainability

Quick Answer: 3 Reasons to Be Cautious About Investing in Environmentally Sustainable Software

Sustainability: A Customer Priority and Provider Imperative

Emerging Technologies: Top Sustainability Trends for Technology and Service Providers