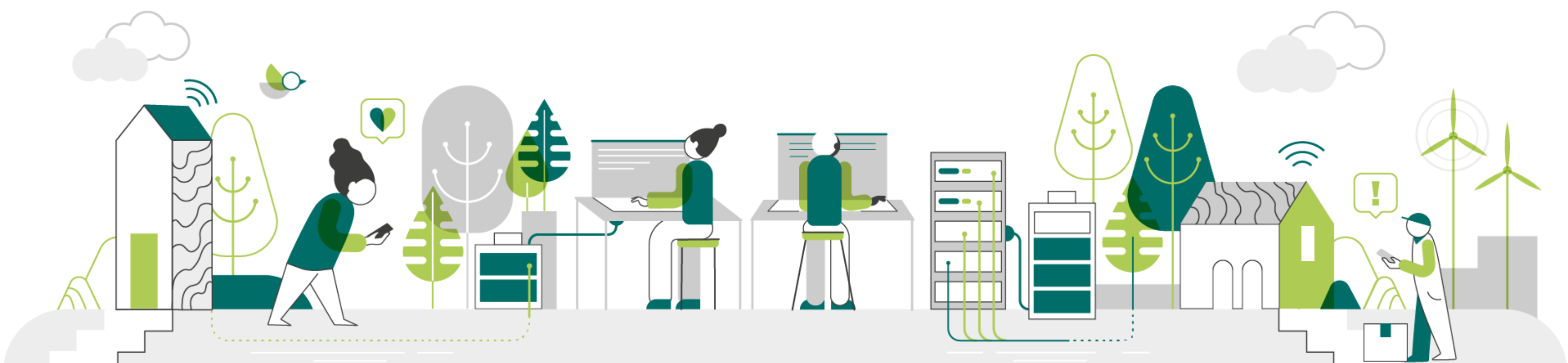
Carbon-aware computing

Measuring and reducing the carbon intensity associated with software in execution

Will Buchanan [a,d], John Foxon [b], Daniel Cooke [b], Sangeeta Iyer [a], Elizabeth Graham [a], Bill DeRusha [a], Christian Binder [a], Kin Chiu [b], Laura Corso [c], Henry Richardson [c], Vaughan Knight [a,d], Asim Hussain [d], Avi Allison [a], Nithin Mathews [a]

[a] Microsoft, 1 Microsoft Way, Redmond, Washington, USA
[b] UBS AG, Bahnhofstrasse 45, 8001 Zurich, Switzerland
[c] WattTime, 490 43rd St, Oakland, California, USA
[d] Green Software Foundation, 3500 South Dupont Highway Suite AA101, Dover, Delaware, USA



About this white paper

The Green Software Foundation (GSF) is a cross-industry consortium that is building a trusted ecosystem of people, standards, tooling, and best practices for “green software.” The vision of green software is to build software that has no harmful effect on the environment: this can be through reducing energy usage, using less hardware, and modifying computation to take advantage of the lowest-carbon sources of energy possible. A core design principle of green software is *carbon-aware computing*, which has been at the core of the cloud partnership between two global organizations, Microsoft and UBS.

This white paper describes how Microsoft, UBS, WattTime, and others partnered through the GSF to develop and release open-source initiatives that advance the state of carbon-aware computing. First, it describes a novel carbon emissions measurement methodology known as the software carbon intensity specification. Next, it presents the GSF’s open-source software development kit named the carbon-aware-sdk, which enables any person or organization to implement carbon-aware computing. Finally, it presents business context and architectural guidance around the implementation of an enterprise-grade carbon-aware application: the UBS core risk platform named Advanced Compute Quantum Analytics (ACQA).

All readers of this white paper should be able to identify the underlying challenges, decisions, and opportunities that arise from implementing carbon-aware capabilities. Business decision makers and product managers should find this white paper valuable to understand and explain the use-cases and principles of carbon-aware computing to key stakeholders. Developers and architects should be able to leverage practical architectural guidance. This whitepaper is intended to help all readers understand the methodology and find the resources necessary to reduce the carbon intensity of their organizations’ software stack.

Revision history

Version	Date	Comments
1.0	January, 2023	Time-shifting MVP spotlighting UBS ACQA application

Problem statement

Environmental sustainability has become a major focus for corporate governance. Regulators from around the globe are increasingly demanding that corporations record, report, and reduce their emissions. Climate governance has already begun to impact supply chains and businesses around the globe.

Software has the potential to accelerate progress towards climate goals. For example, software can be used to drive digital transformation that helps reduce physical movement of people or goods, or to support decarbonization efforts across industries. However, rapid growth in software technologies and democratization through cloud services has provided organizations with unprecedented access to computational power, which could incur high energy usage and associated emissions costs. Hence, to accelerate our technological progress in a sustainable manner, software itself needs the ability to measure and reduce its impact on carbon emissions.

Moving to the cloud is a key part of many organizations' decarbonization efforts through centralized management, consolidation of resources, and efficiency improvements. However, datacenters and data transmission networks currently account for nearly 1% of energy-related global greenhouse gas emissions, and will very likely increase in the near future. As organizations increasingly migrate to the cloud, this usage of datacenters will represent a meaningful portion of their carbon footprint. Therefore, datacenters - and the software running on them - can play a meaningful part in decarbonization efforts.

To address this as part of their net zero and carbon negative pledges, major cloud providers, including Microsoft, are already matching their cloud energy consumption with carbon-free energy through market-based neutralization measures such as Power Purchase Agreements (PPAs). PPAs are a powerful tool because they enable the development of new renewable energy projects that help decarbonize the grid. However, *purchasing* clean energy is not the same as *physically consuming* clean energy. Due to the interconnected nature of the electric grid, there is no way to physically allocate generator-specific electricity to a specific energy consumer.

The emissions impact of electricity consumption is thus a function of the overall mix of electricity generation resources on a grid, rather than just those resources from which a consumer purchases electricity. Factoring in the mix of resources on the grid provides additional opportunities for decarbonization. For example, organizations have the opportunity to reduce emissions by not only *investing* in market-based neutralization measures that increase carbon-free supply, but also *managing loads* in ways that reduce emissions. One way that this can be accomplished is by shaping the demand that software itself makes on the grid: this is a core focus of the Green Software Foundation.

The GSF defines *carbon-awareness* as modifying computation to take advantage of the lowest-carbon sources of energy possible to have minimal environmental impact. More specifically, there are opportunities to shift software workloads that require heavy compute power to *times* and *places* where the carbon intensity of the grid will result in the lowest possible carbon emissions.

Every electricity grid incorporates a mix of energy sources that respond to changes as demand (electricity load) varies over time, as illustrated in Figure 1. At certain times and places, increased electricity demand will be met by carbon-free resources such as solar and wind. In other circumstances, increased demand will be met by carbon emitting resources such as coal and natural gas.

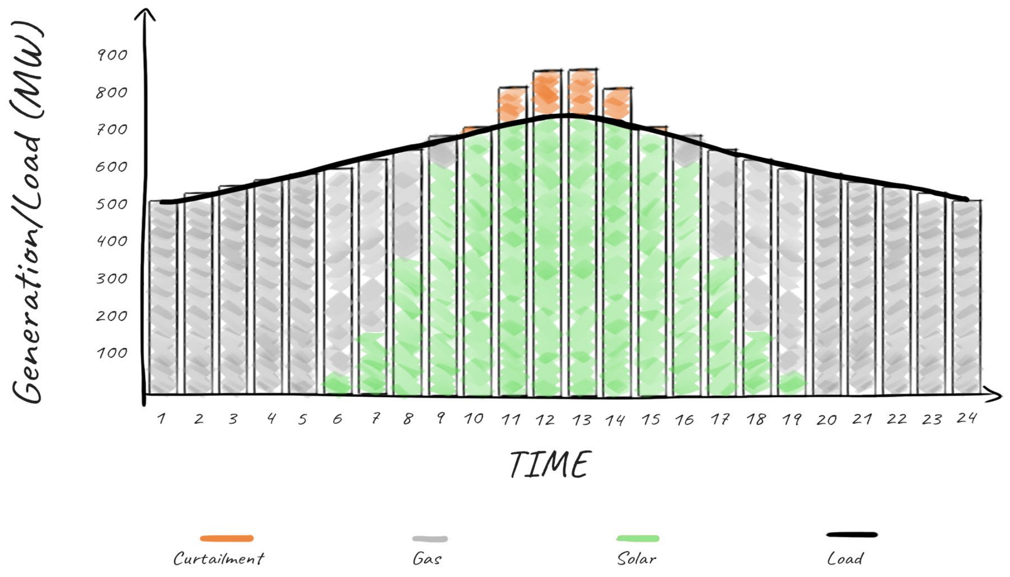


Figure 1 *Visualization of supply and demand over time for a grid with a mixed generation stack. In this example, by shifting load to mid-day, lower carbon resources can respond. For example, there are times in California where there is an oversupply of solar power, and clean electricity is being discarded (or 'curtailed';) consuming more electricity at these times would cause no additional emissions.*

When making decisions to decarbonize software, it is therefore imperative to factor in the *"carbon intensity"* of a grid, defined as the amount of emissions of carbon dioxide (CO_2eq) released per unit of another variable. Modeling the impact of a decision to consume electricity at a given moment or location requires "mar*ginal carbon intensity*" data, which is a measure of the additional carbon emissions that would be created by placing additional demand on a particular electric grid at a specific point in time.

"The way we design and build our applications has a direct relationship to how much carbon they emit. With a better understanding of the impact our application designs have on the environment, we can make choices which have a more positive impact on the planet."

— Paul McEwen, Global Head of Technology Services for UBS

These marginal carbon intensity emissions factors must be spatially and temporally specific, requiring granular information about what generating resource is responding to a change in load so that the cleanest times and locations can be selected. Better marginal emissions data is essential for identifying the exact times and locations to enable carbon-aware software to cause the least emissions possible.

Cloud providers and their customers can partner to reduce emissions: there are certain carbon-aware decisions that only customers or end users can implement to make their workloads less carbon-intensive, such as choosing a geographic region, or deciding when to run a workload. To make informed choices, users require transparency from cloud providers to understand what actions are available, and accurately assess the impact of their decisions.

"At Microsoft, we are committed to helping organizations reach their sustainability goals. Carbon-aware computing advances the measurement and reduction of carbon emissions associated with software technology estates at a global scale."

— Elisabeth Brinton, Microsoft Corporate Vice President, Sustainability

To make software carbon-aware, organizations must have ready access to the information and tooling required to drive implementation. While major software platforms such as Windows 11, Xbox, Google Cloud Platform, and Apple iOS have already implemented proprietary carbon-aware functionalities that reach billions of devices and users, open-source standards and tools were not yet available to the software industry. The impacts of a carbon-aware decision – both forecasted and actual – need to be measured and reported. This requires sufficient telemetry and reporting protocols.

Current corporate reporting standards and practices, however, do not adequately incentivize usage of carbon-aware design patterns. The most broadly adopted standard is the Greenhouse Gas Protocol (GHGP); under current GHGP reporting practices electricity emissions are generally calculated using broad (regional, annual) average emission rates. Such emissions rates are insufficiently granular on a spatial and temporal basis and do not empower green software decisions. Carbon reporting tools such as Microsoft's Emissions Impact Dashboard align with the GHGP, and therefore will not yet reflect carbon-aware decisions implemented by their customers.

There is an opportunity for protocols such as the GHGP to be updated to better reflect the emissions impact of carbon-aware computing, renewable energy procurement, and other electric-sector decisions. In the interim, cloud providers and organizations can immediately take actions to measure and reduce their software's carbon intensity.

Solution statement

A core principle of carbon awareness is commonly referred to as *time-shifting* and is illustrated in Figure 2 as the focus of this white paper. An enterprise-grade carbon-aware minimum viable product (MVP) is presented that solely relies on open-source tooling co-developed and made available to the public by multiple organizations under the GSF umbrella. The MVP focuses on the UBS owned core risk platform named Advanced Compute Quantum Analytics (or ACQA -- pronounced aqua or /ˈɑː.kwə/) on which Azure Batch based workloads are shifted to *times* with lower marginal carbon intensity within a 24 hour window.

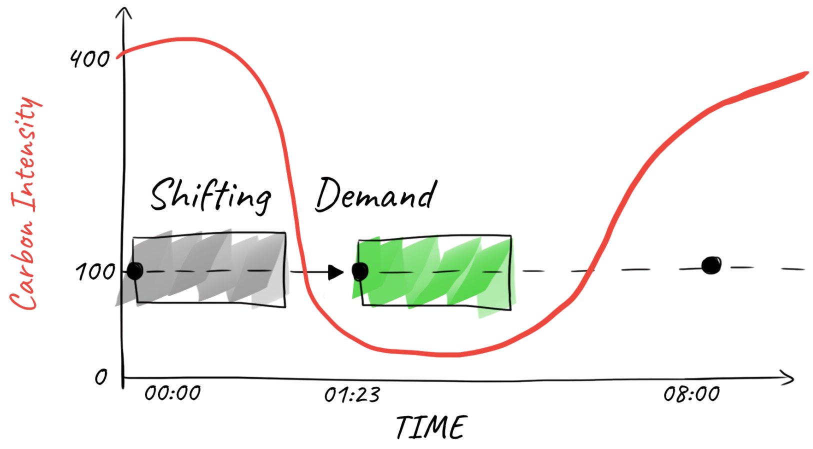


Figure 2 *Visualization of time-shifting a workload from an "on-demand" time of high carbon intensity to a later time with lower carbon intensity.*

A previous publication by Microsoft identified an average potential of 15% reduction in software carbon intensity (SCI) across 16 different regions for time-shifting. As the study showed, the highest carbon-aware reductions are to be gained from time-shifting computationally-intensive workloads with short durations and high volumes that reside in regions with high renewable energy intermittency. Therefore, key MVP selection criteria for a UBS carbon-aware application were *computational intensity*, *short durations (30 minutes or less)*, *high-volumes*, and *deferrable* start times. The expected SCI reductions based on Microsoft's previous study were corroborated through empirical observations of ACQA's historical data and a European grid mix. When deployed at scale, carbon-aware ACQA can lead to the avoidance of many metric tons of CO_2eq emissions per annum.

To address the problem of insufficient measurement and reporting standards, Microsoft, UBS, WattTime, and other GSF members contributed to the development and release of the GSF's Software Carbon Intensity specification. This specification factors in the marginal carbon intensity of the grid, the energy consumed by the software, and the embodied carbon of the devices themselves. The SCI is a rate of carbon emissions per unit of 'R', which is a functional unit of scale relevant to the software being measured. For some systems, R might be the number of users; for others it might be per workload. The SCI is based on the carbon intensity of the grid itself, and is therefore not possible to neutralize a SCI score through market-based mechanisms.

The equation used to calculate the SCI value of a software system is:

```
SCI = ((E * I) + M) per R
```

where,

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., carbon per additional user).

With these variables in hand, Microsoft and UBS partnered through the GSF to build a carbon-aware MVP. For feasibility and impact reasons, time-shifting was identified as the MVP scope hosted by the GSF's carbon-aware-sdk (see an overview shown in Figure 3). Although beyond the MVP scope, the teams have also implemented location-shifting functionalities into the SDK: Microsoft's previous research has demonstrated that choosing an appropriate region can have the *largest* SCI reduction impact (almost 75%). The SDK also meets reporting requirements and provides measured carbon intensity for time periods when batch jobs have already run and recommends execution times based on marginal carbon intensity data forecasts.

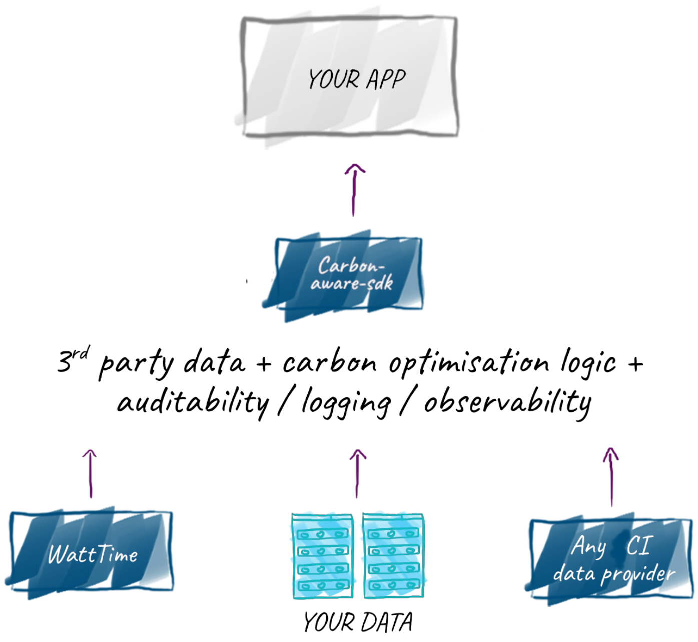

YOUR APP
Carbon-aware-sdk
3rd party data + carbon optimisation logic + auditability / logging / observability
WattTime
YOUR DATA
Any CI data provider

Figure 3 *A schematic showing how the carbon-aware-sdk interfaces with carbon intensity (CI) data providers to offer carbon-aware functionalities to an application.*

The carbon-aware-sdk is interoperable with different carbon intensity data providers. At its core, the carbon-aware-sdk is a Web API and Command Line Interface (CLI), with identical functionalities. The API can be deployed as a container for easy management, and can be deployed alongside an application within a cluster or separately.

The API consistently enables informed decisions to be made through recommendations for both *time*- and *location*-shifting, as well as enabling telemetry capture for reporting. Through WattTime, a temporal resolution of 5-minute intervals is provided and can allow historical data or forecast within a 24-hour window to be queried. Currently, it can be queried to provide carbon intensity information for more than sixty global regions of Microsoft Azure's cloud computing platform. Locations of interest can also be provided through a query – all queries are return JSON objects.

The carbon-aware-sdk both reports historical marginal carbon intensity and provides a forecast. Therefore, the SDK can also be used to model the impact of potential time-shifting decisions that may result from adapting a time-shifting strategy before integrating with applications. In summary, the SDK allows applications to consume more renewable energy and less fossil-fuel-based energy by recommending optimal times and locations.

"By simply building in the intelligence to know when and where electricity consumption will have low — or even zero — carbon impact, carbon-aware software can become a powerful tool to drive the energy transition forward."

— Gavin McCormick, Founder and Executive Director, WattTime

An MVP focusing on time-shifting

Both *time*- and *location*-shifting are relevant carbon-aware strategies for deferrable workloads. There can be, however, additional complexities associated with location-shifting, such as the ancillary emissions associated with data-transfer and cloud resource provisioning, as well as region sovereignty and GDPR considerations. For the MVP, the teams decided to focus on time-shifting workloads. In what follows, a four-step case study is presented (see Figure 4), illustrating the methodology that UBS and Microsoft applied to model the carbon-aware-sdk, and to validate the impact of time-shifting efforts ahead of executing Azure Batch workloads from ACQA.

Step 1: Measure carbon intensity of a past workload
Consider an example where an Azure Batch job ran for 15 minutes at 10:00 am on September 19, 2022, in the Central US Azure datacenter. These details can be sent to the carbon-aware-sdk to calculate the mean marginal carbon intensity of that job.

```
<SDK-API-SERVER>/emissions/mean-marginal-carbon-intensity
     location=centralus
     startTime=2022-09-19T10:00:00+00:00
     endTime=2022-09-19T10:15:00+00:00
```

For this query, the carbon-aware-sdk returns a mean marginal carbon intensity of 719 gCO2eq/kWh.

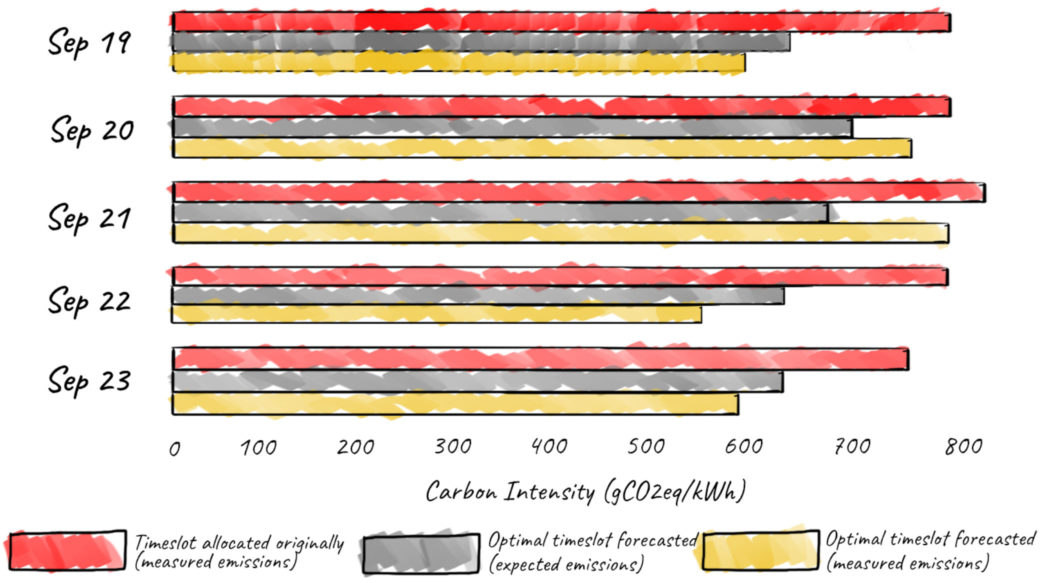


Figure 4 *Illustration of potential savings modeled for an example Azure Batch job for an entire work week. Historical data (red) is compared with both forecasted (grey) and actual (yellow) data to determine real-world SCI reduction.*

Step 2: Determine optimal forecasted carbon intensity
Next, the carbon-aware-sdk is queried for what the 15-minute-average marginal carbon intensities were forecasted to be on September 19.

```
<SDK-API-SERVER>/forecasts/batch
     requestedAt: 2022-09-19T00:00:00+00:00
     location: centralus
     windowSize: 15
```

In response, the carbon-aware-sdk returns 15-minute-averages for the entire day at 5-minute increments (00:00 to 00:15, 00:05 to 00:20, 00:10 to 00:25, etc.) and the specific optimal 15-minute window with the lowest carbon intensity. In this case, it is 659 gCO2eq/kWh at 7:05 to 7:20 AM.

Step 3: Measure carbon intensity of that optimized workload
The optimal carbon intensity value of 659 gCO2eq/kWh is a forecast. To determine the actual carbon savings this potential time-shift would have captured, the actuals from 7:05 AM need to be acquired (Step 1), but with a new window (7:05 to 7:20 AM.)

```
<SDK-API-SERVER>/emissions/mean-marginal-carbon-intensity
     location=centralus
     startTime=2022-09-19T07:05:00+00:00
     endTime=2022-09-19T07:20:00+00:00
```

Step 4: Iterate steps 1 through 3 to identify the potential savings over time
Our query shows the difference between historical (719) forecast (~659) and measured (~642). For this one day, steps 1 to 3 indicate that the potential impact of time-shifting a 15-minute Azure batch job to the optimal time could result in a 10% reduction in the SCI. Repeating the above steps iteratively, the expected impact of time-shifting on the single job can be calculated as the carbon savings accumulate day to day, week to week, and month to month. Analysis of 6 months of historical data revealed savings

that were closely in alignment with similar workloads previously published research by Microsoft, providing a strong signal supporting the integration of carbon-aware-sdk into UBS ACQA risk platform.

The only difference between the modeling exercise above and a live scenario is the order of operations. ACQA queries the most recent forecast data and optimal time and schedules the job for that time: after the job completes, ACQA reports on the measured outcomes.

MVP architecture rendering ACQA carbon-aware

The carbon-aware-sdk has been integrated into the ACQA risk platform. For each job executed on Azure batch, it is being used to capture both actual and forecast optimal marginal carbon intensity. Please refer to Figure 5 for a schematic overview.

The ACQA management information system (MIS) service sources information describing each job from Application Insights' custom telemetry records and then queries the carbon-aware-sdk via REST to find the intensity calculations. The intensity values are then ingested into the Databricks ACQA data lake and made available alongside other MIS data such as compute cycles. Databricks, when combined with tools such as Power BI, allow targeted reports to be generated – for example listing the top 20 jobs with the largest workloads being run at the least optimal times. Currently marginal carbon intensity data is being used for observation only. The next phase will be to integrate the optimal forecast into the risk platform scheduling process to delay the execution of jobs. For non-time sensitive jobs, for example back-

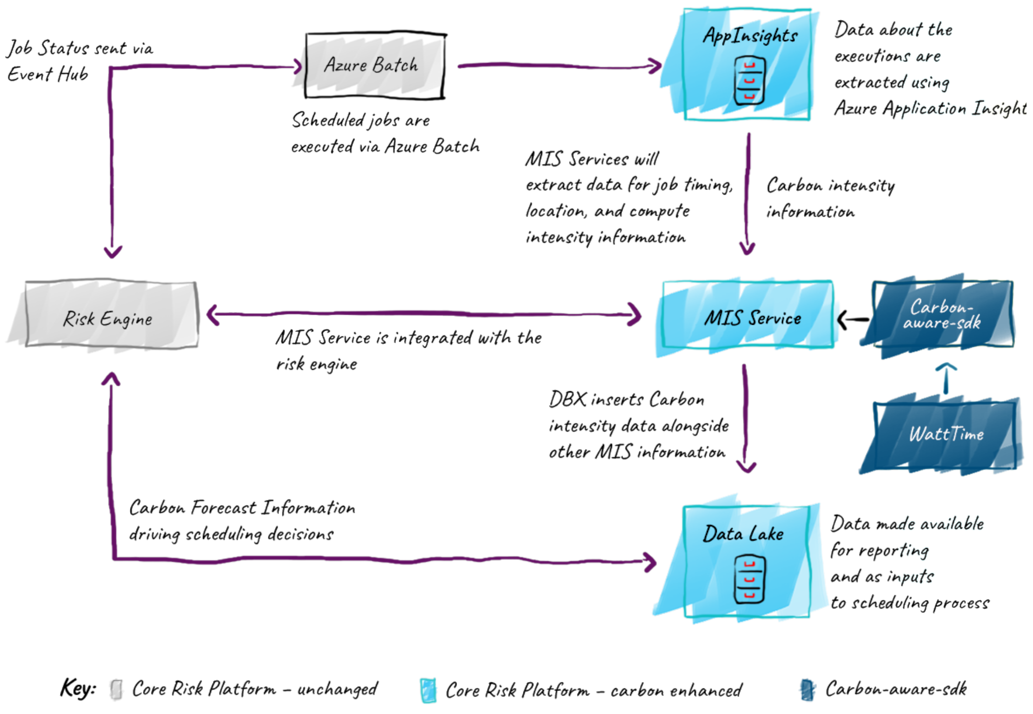

Job Status sent via Event Hub
Azure Batch
Scheduled jobs are executed via Azure Batch
AppInsights
Data about the executions are extracted using Azure Application Insight
MIS Services will extract data for job timing, location, and compute intensity information
Carbon intensity information
Risk Engine
MIS Service is integrated with the risk engine
MIS Service
Carbon-aware-sdk
WattTime
DBX inserts Carbon intensity data alongside other MIS information
Carbon Forecast Information driving scheduling decisions
Data Lake
Data made available for reporting and as inputs to scheduling process
Key: Core Risk Platform – unchanged
Core Risk Platform – carbon enhanced
Carbon-aware-sdk

Figure 5 *Architecture depicting ACQA's integration with the carbon-aware-sdk*

testing of risk scenarios, this will be trivial to achieve. When jobs have deadlines, such as risk reports for a trading desk, any time-shifts will have to ensure that the job is complete by its required time. Any time-shifting will also need to consider the maximum capacity of the cloud infrastructure; some jobs will need to run at non-optimal times to efficiently utilize infrastructure capacity.

Future opportunities

Green Software Engineering is presented with some hard technical challenges. Key opportunities arise between cloud providers and customers to implement carbon-aware design patterns and increase accuracy and coverage of emissions data:

Full SCI score
This work aspires toward reporting a full Software Carbon Intensity score; the journey to carbon-awareness requires a crawl/walk/run approach to fulfill the SCI equation (SCI = ((E * I) + M) per R).

- **Crawl:** First, carbon intensity (*I*) is the required metric to begin time-shifting workloads. The MVP's success will be measured as a relative percentage of reduction in SCI based on the reduction in marginal carbon intensity. There remains an opportunity for standardization and improvement in the granularity, accuracy, and transparency of marginal emissions rate data and methodologies used for carbon-aware decision making.
- **Walk:** Next, to capture the absolute impact, the energy consumption (*E*) needs to be incorporated. To start, a heuristic must be developed to estimate energy consumption for a virtual machine or docker container. Eventually, this will be replaced with accurate energy telemetry; Microsoft has already made this available for GPU energy consumption in Azure Machine Learning.
- **Run:** Finally, the embodied carbon ('M') of the physical resources required to run the computation will need to be integrated. It is imperative that cloud providers make this information accessible to customers.

Location selection
Workloads can be provisioned and/or executed in different regions (and thus on different electricity grids) based on which one has lower carbon intensity. Such a decision can have lasting effects over the lifetime of the resource. The gains are potentially much higher: Microsoft recently partnered with leading academic researchers to demonstrate the SCI can be reduced by up to 75% by choosing the appropriate location. This requires some enhancements to the API to account for additional carbon emission factors (e.g. data transfer).

Energy telemetry
It is challenging to directly and accurately measure power consumption for Virtual Machines, especially in the cloud. Heuristic models can be used to estimate energy consumption based on proxies such as utilization. Cloud and hardware providers may invest in increasing accuracy and providing transparency to customers.

Conclusions

There is a clear industry-wide demand for advancements in the "access to" and "accuracy of" software carbon intensity data. Microsoft, UBS, WattTime, the Green Software Foundation, and others are driving adoption of standards such as the SCI and open-source tooling such as the carbon-aware-sdk. This shift towards accessible, accurate, real-time, and granular software carbon intensity can unlock incredible opportunities both within the boundaries of software industry and beyond. Carbon-aware design patterns are relevant in any grid-based application; adoption at a global scale can expedite our societies' decarbonization ambitions.

This white paper presented an enterprise-grade implementation of carbon-aware computing using the SCI specification and carbon-aware-sdk. The artifacts developed through the MVP can be leveraged by organizations looking to reduce carbon emissions by *time*- or *location*-shifting their workloads. Presently, UBS is in the process of ramping up adoption of their core risk platform workloads to deploy an automated carbon-aware job scheduler. This adoption has the potential to decrease ACQA's SCI by 15%, as identified by Microsoft's previous research on similar workloads. At the scale of ACQA workloads, this single optimization based on the co-developed MVP would prevent many metric tons of potential CO2eq from entering the atmosphere every year.

How to get involved

Organizations and software engineers are invited to consider carbon-aware computing as a viable option when building new software solutions or when modernizing existing ones. They are therefore invited to contribute to and leverage the GSF's carbon-aware-sdk or SCI specification, The GSF organizes an annual hackathon for carbon-aware computing; read about the great ideas that emerged from CarbonHack22, and dig deeper into green software principles here.

To find out more about the GSF, its members and ongoing activities and events, please visit https://greensoftware.foundation/.

There is finally a way for software to be more energy-responsive based on supply and demand. Developers and organizations looking towards sustainability and utilizing clean energy should lean on the Green Software Foundation and the tools now available to decarbonize their software and meet their sustainability targets.

— Asim Hussain, Executive Director & Chairperson, Green Software Foundation

Key definitions

Carbon-awareness: The Green Software Foundation defines this as doing more when more energy comes from low carbon sources and doing less when more energy comes from high carbon sources. Carbon awareness changes the behavior of an application to take advantage of clean, renewable or low carbon sources of electricity, with the following principles:

- **Time-shifting:** Optimize impact by shifting workloads based on the Location-Based Marginal Carbon Intensity prediction of the datacenter
- **Location-shifting:** Optimize impact by shifting workloads based on the Location-Based Marginal Carbon Intensity prediction of a location
- **Demand-shaping:** Running software so it does more when electricity is clean and less when it's dirty

Carbon-aware-sdk: The Green Software Foundation has created an SDK to enable the creation of carbon aware applications, applications that do more when the electricity is clean and do less when the electricity is dirty

Carbon efficiency: Changes the software / architecture of an application so that it is responsible for emitting less carbon

Energy efficiency: Using less energy to do the same work

Hardware efficiency: Using less hardware to do the same work

Carbon intensity: The amount of emissions of carbon dioxide (CO_2eq) released per unit of another variable; in our case, energy consumption.

Average carbon intensity: Average carbon emissions are calculated using the total carbon emissions and total amount of electricity generated. This average is taken across all generators

Marginal carbon intensity: The carbon intensity of electricity is a measure of how much carbon (CO_2eq) emissions are produced per kilowatt-hour (kWh) of electricity consumed; this is based on the grid that provides the electricity. This is a measure of the emissions intensity of the marginal power plant(s) that will respond to your change in electricity demand (i.e., scheduling compute load)

SCI equation: The equation used to calculate the SCI value of a software system is SCI = ((E * I) + M) per R, where:

- E = Energy consumed by a software system
- I= Location-based marginal carbon emissions (WattTime for Azure Regions)
- M = Embodied emissions of a software system
- R = Functional unit (Azure Compute Job)

Software Carbon Intensity (SCI) specification: The Software Carbon Intensity technical specification describes how to calculate the carbon intensity of a software application. It describes the method of calculating the total carbon emissions and the selection criteria to turn the total into a rate that can be used to achieve real-world, physical emissions reductions, also known as abatement

Acknowledgements

This work is a culmination of ongoing efforts from many individuals across the software industry who have contributed to the foundational and technical development, provided inspiration and guidance, or sponsored crucial activities. This includes, but is not limited to: Abhishek Gupta, Akshara Ramakrishnan, Alex Bitukov, Bill Johnson, Chandrika Jain, Conor Kelly, Dan Balma, Dan Taylor, Daniel Moth, David Collier, Dorota Zimnoch, Emma Strubell, Erica Basham, Gavin McCormick, George Lara-Matthews, Holly Beale, Jason Oppler, Jennifer Madiedo, Jesse Dodge, John Mymryk, Juan Zuluaga, Kanishk Tantia, Mehrnoosh Sameki, Nabila Babar, Priti Pathak, Ravi Varma Addala, Remi Tachet des Combes, Robert May, Rose Brennan, Roy Schwartz, Sasha Luccioni, Scott Chamberlain, Sharon Gillett, Srilatha (Bobbie) Manne, Steve Sweetman, Tammy McCllellan, Taylor Prewitt, and Terry Beale.

Contributors

Microsoft (NASDAQ "MSFT" @microsoft) enables digital transformation for the era of an intelligent cloud and an intelligent edge. Its mission is to empower every person and every organization on the planet to achieve more.

UBS (NYSE: "UBS" @UBS Group AG) convenes the global ecosystem for investing, where people and ideas are connected and opportunities brought to life, and provides financial advice and solutions to wealthy, institutional and corporate clients worldwide, as well as to private clients in Switzerland. UBS offers investment solutions, products and impactful thought leadership, is the leading global wealth manager, provides large-scale and diversified asset management, focused investment banking capabilities, and personal and corporate banking services in Switzerland. The firm focuses on businesses that have a strong competitive position in their target markets, are capital efficient and have an attractive long-term structural growth or profitability outlook. UBS is present in all major financial centers worldwide. It has offices in more than 50 regions and locations, with about 30% of its employees working in the Americas, 30% in Switzerland, 19% in the rest of Europe, the Middle East and Africa and 21% in Asia Pacific. UBS Group AG employs more than 72,000 people around the world. Its shares are listed on the SIX Swiss Exchange and the New York Stock Exchange (NYSE).

WattTime is an environmental tech nonprofit that empowers all people, companies, policymakers, and countries to reduce emissions and choose cleaner energy. Founded by UC Berkeley researchers, they develop data-driven tools and policies such as Automated Emissions Reduction (AER), software that allows for computing to run on cleaner energy and Emissionality, which enables siting new renewables based on the greatest avoided emissions impact.

The Green Software Foundation is building a trusted ecosystem of people, standards, tooling and best practices for Green Software. It has been established as the Joint Development Foundation Projects, LLC, Green Software Foundation Series (the "Project"). The Joint Development Foundation is a non-profit organization that provides the corporate and legal infrastructure to enable groups to establish and operate standards and source code development collaborations. The Joint Development Foundation is an affiliate of the Linux Foundation.