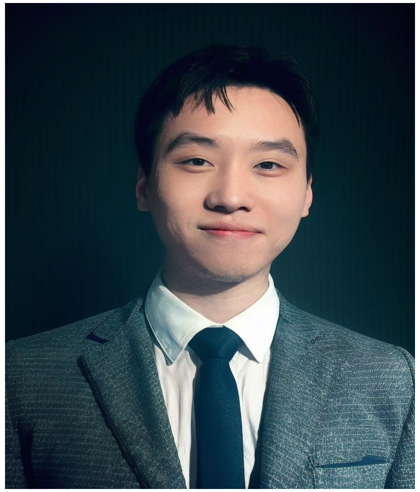
The Hidden Carbon Footprint of Serverless Computing



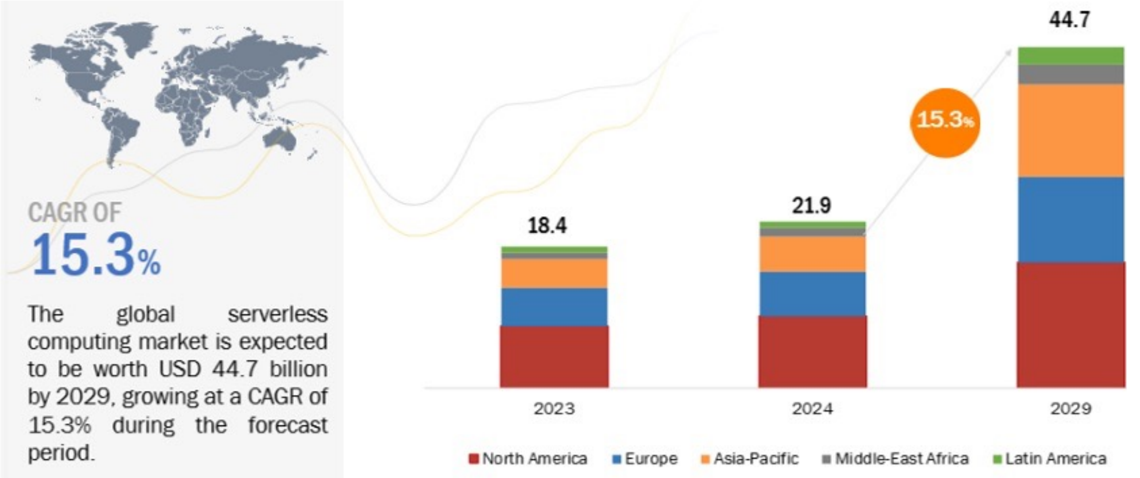
Rohan Basu Roy

Raghavendra Kankagiri

Yankai Jiang

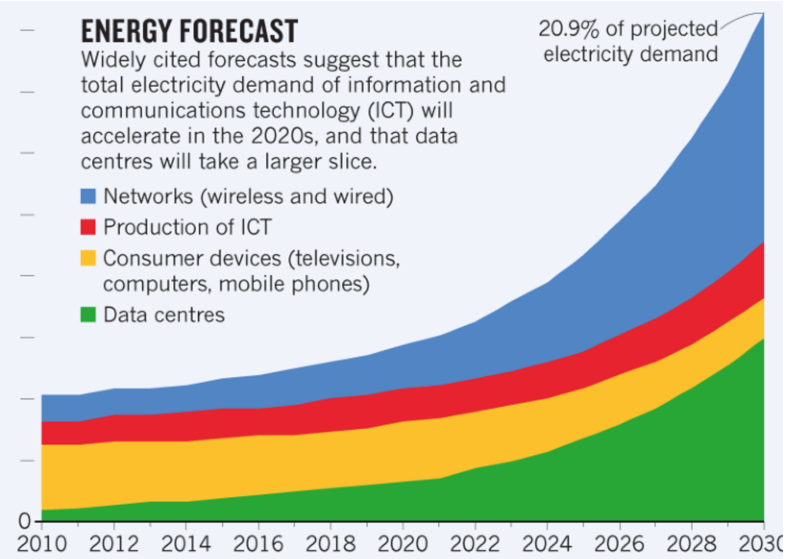
Devesh Tiwari

Why are we interested in this problem?



* sourced from https://www.datadoghq.com/state-of-serverless/

More than 70% of users of production cloud platforms like AWS, using one or more serverless solutions

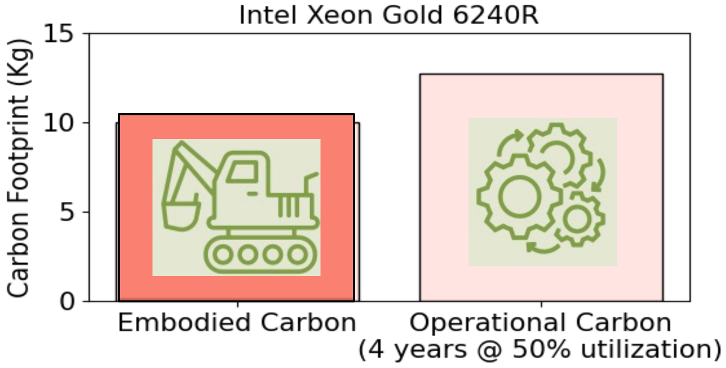


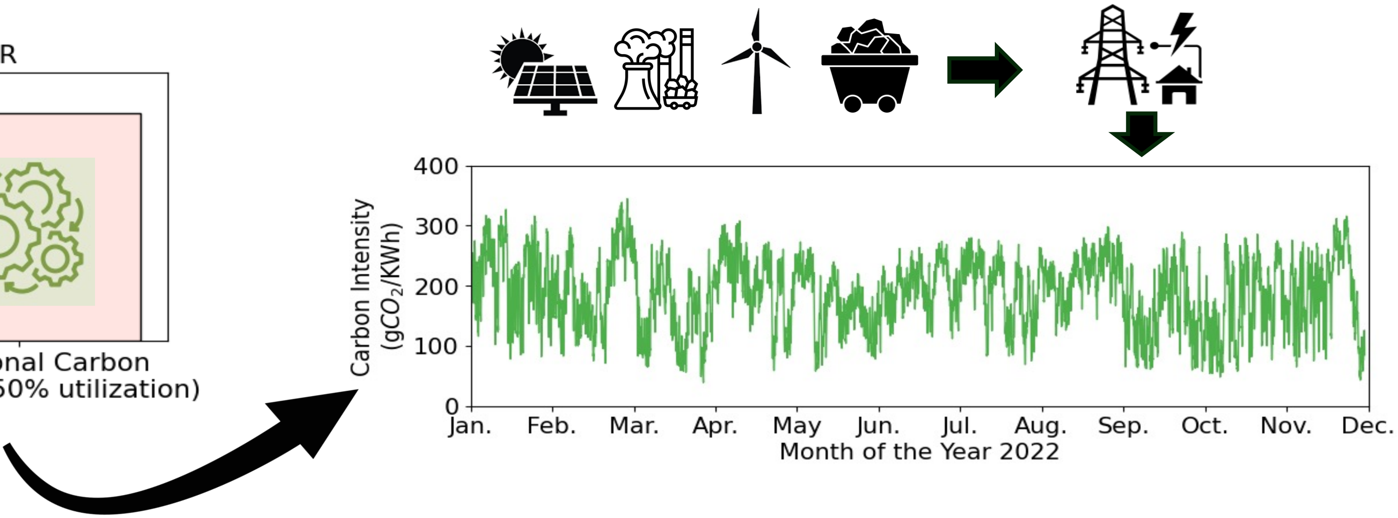
* sourced from https://www.nature.com/articles/d41586-018-06610-y

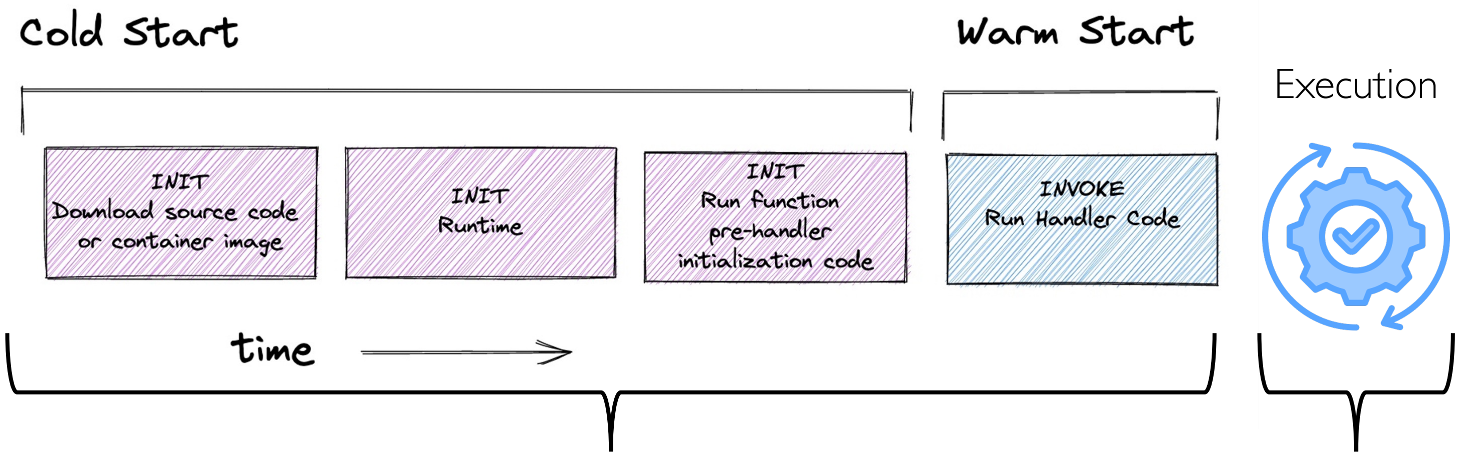
The energy demand, and the carbon footprint of datacenters is projected to grow at a rapid rate

Why is carbon accounting for serverless challenging?

Embodied carbon changes with hardware and operational carbon changes with carbon intensity







Keep-alive carbon

Execution carbon

Sharing resources among functions during execution and keep-alive make carbon accounting challenging

Accounting for serverless carbon footprint is challenging and methodologically error-prone because of the unique keep-alive period aspects of serverless functions

Keep-alive carbon has operational and embodied components

DRAM Embodied Carbon

Keep-alive Time

Carbon Intensity

$$C_{f_{\text{keep-alive}}} = \underbrace{\frac{E_{\text{DRAM}}}{L_{\text{DRAM}}} * T_{\text{keep-alive}}}_{\text{Embodied Carbon}} + \underbrace{\text{CI} * J_{\text{DRAM}}}_{\text{Operational Carbon}}$$

Choice A

DRAM Lifetime

DRAM Energy

DRAM's embodied carbon has a significant environmental impact. This assumes that only one function is running/ kept alive in a server.

Keeping functions alive on harvested memory improves performance and should be included in carbon calculation

Fraction of memory consumed by function

$$C_{f_{\text{keep-alive}}} = \frac{E_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}{L_{\text{DRAM}}} * T_{\text{keep-alive}} + \text{CI} * J_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}$$

Choice B

Embodied Carbon

Operational Carbon

But CPU cores should remain reserved during keep-alive so that functions can run when invoked. Where is their carbon?

Reservation of CPU cores during keep-alive period consumes embodied carbon

Fraction of cores reserved

$$\underbrace{C_{f_{\text{keep-alive}}}}_{\text{Choice C}} = \Big(\underbrace{\frac{E_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}{L_{\text{DRAM}}}}_{\text{Embodied DRAM}} + \underbrace{\frac{E_{\text{CPU}} * \overbrace{\frac{n_f}{N_{\text{CPU}}}}^{\text{Fraction of cores reserved}}}{L_{\text{CPU}}}}_{\text{Embodied CPU}} \Big) * T_{\text{keep-alive}} + \underbrace{\text{CI} * J_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}_{\text{Operational DRAM}}$$

The cores still consume idle power even when they are not actively executing functions during the keep-alive period

Keep-alive carbon should include the operational and embodied carbon of DRAM and CPU

Embodied DRAM

Embodied CPU

$$C_{f_{\text{keep-alive}}} = \left(\boxed{\frac{E_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}{L_{\text{DRAM}}}} + \boxed{\frac{E_{\text{CPU}} * \frac{n_f}{N_{\text{CPU}}}}{L_{\text{CPU}}}} \right) * T_{\text{keep-alive}} +$$

Choice D

$$\text{CI} * \left(\boxed{J_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}} + \boxed{J_{\text{CPU}} * \lambda_{\text{IDLE}} * \frac{n_f}{N_{\text{CPU}}}} \right)$$

Operational CPU

Operational DRAM

Idle power factor

Since CPU cores are indirectly consumed, their carbon accounting is harder than DRAM due to varying implicit reservations across providers

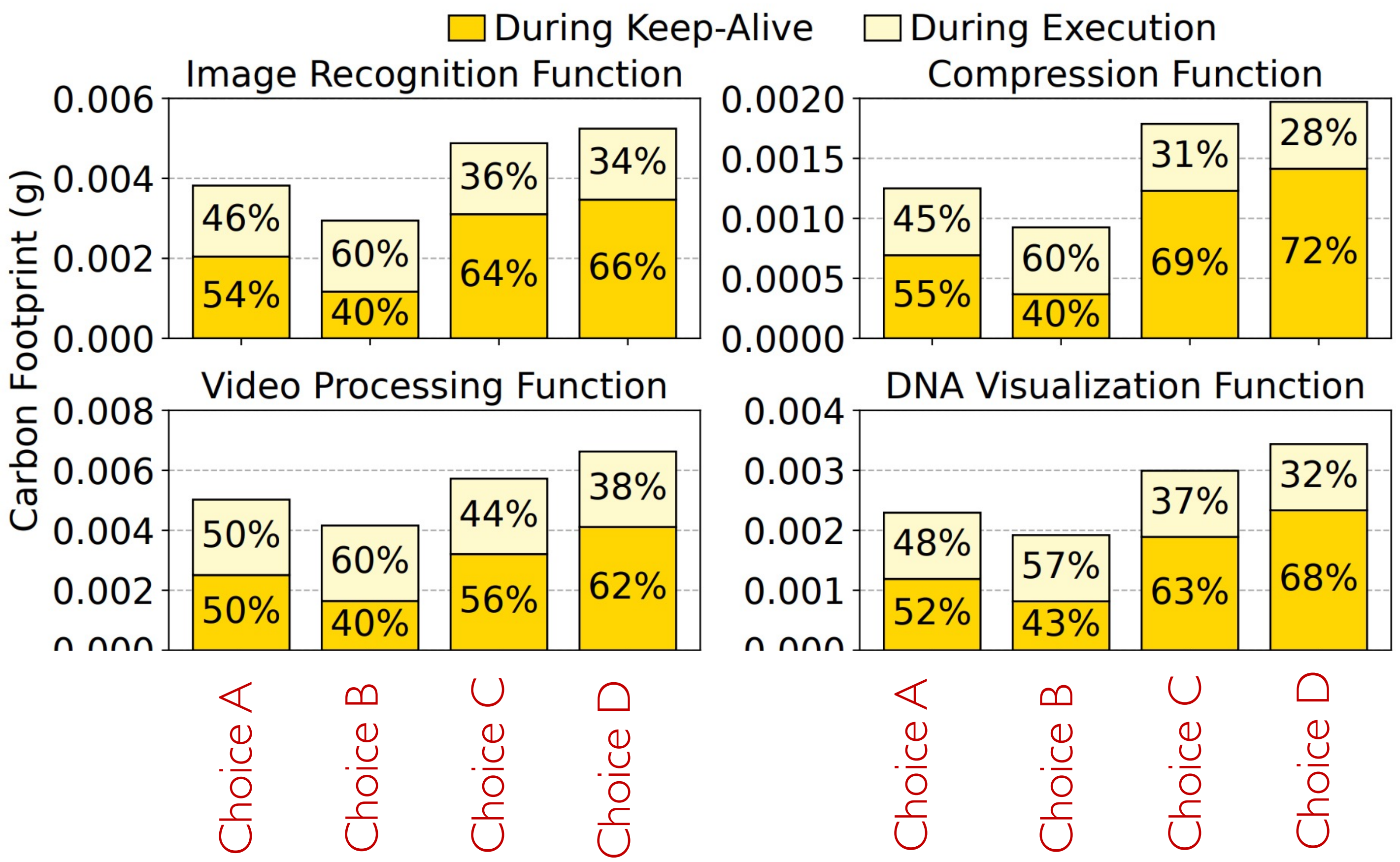
Functions lack fixed hardware, making it harder to estimate carbon impact across diverse systems

$$C_{f_{\text{execution}}} = \left(\underbrace{\frac{E_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}{L_{\text{DRAM}}}}_{\text{Embodied DRAM}} + \underbrace{\frac{E_{\text{CPU}} * \frac{n_f}{N_{\text{CPU}}}}{L_{\text{CPU}}}}_{\text{Embodied CPU}} + \underbrace{\frac{E_{\text{ST}} * \frac{S_f}{S_{\text{ST}}}}{L_{\text{ST}}}}_{\text{Embodied Storage}} \right) * T_{\text{exe}} + \text{CI} * \left(\underbrace{J_{\text{DRAM}} * \Delta_{f_{\text{DRAM}}}}_{\text{Operational DRAM}} + \underbrace{J_{\text{CPU}} * \frac{n_f}{N_{\text{CPU}}}}_{\text{Operational CPU}} + \underbrace{J_{\text{ST}} * \frac{S_f}{S_{\text{ST}}}}_{\text{Operational Storage}} \right)$$

The short runtime and low resource needs of serverless functions make them ideal for co-location, but this complicates accurate carbon accounting, requiring precise energy measurement

What are the implications of different methodological choices and scheduling on the serverless carbon footprint?

Keep-alive outweigh execution in carbon due to brief executions and extended keep-alive periods



Choice A: DRAM embodied and operational for keep-alive carbon

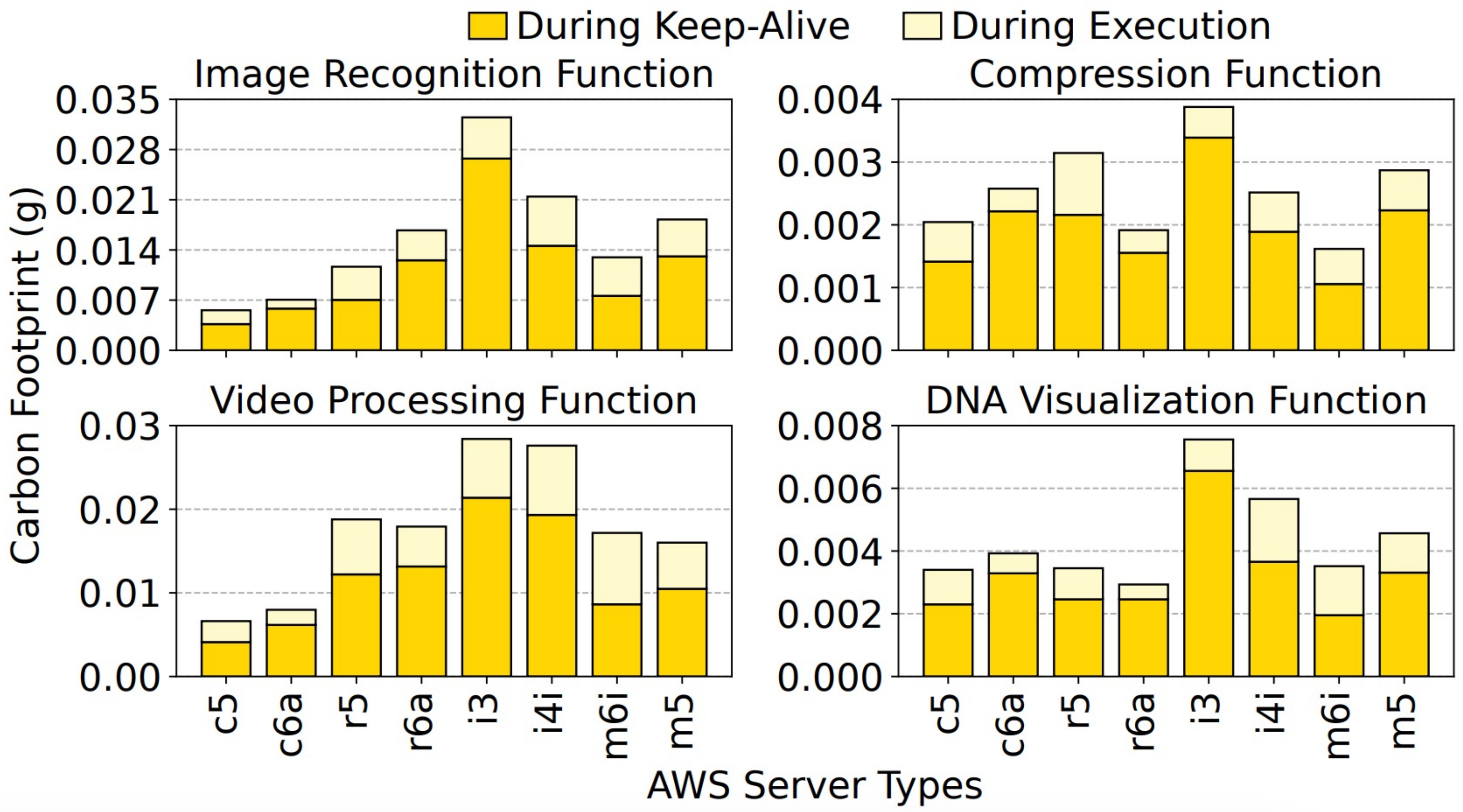
Choice B: DRAM embodied and operational for keep-alive carbon, divided among co-located functions

Choice C: DRAM embodied and operational for keep-alive carbon, CPU embodied for keep-alive

Choice D: DRAM embodied and operational for keep-alive carbon, CPU embodied and operational for keep-alive

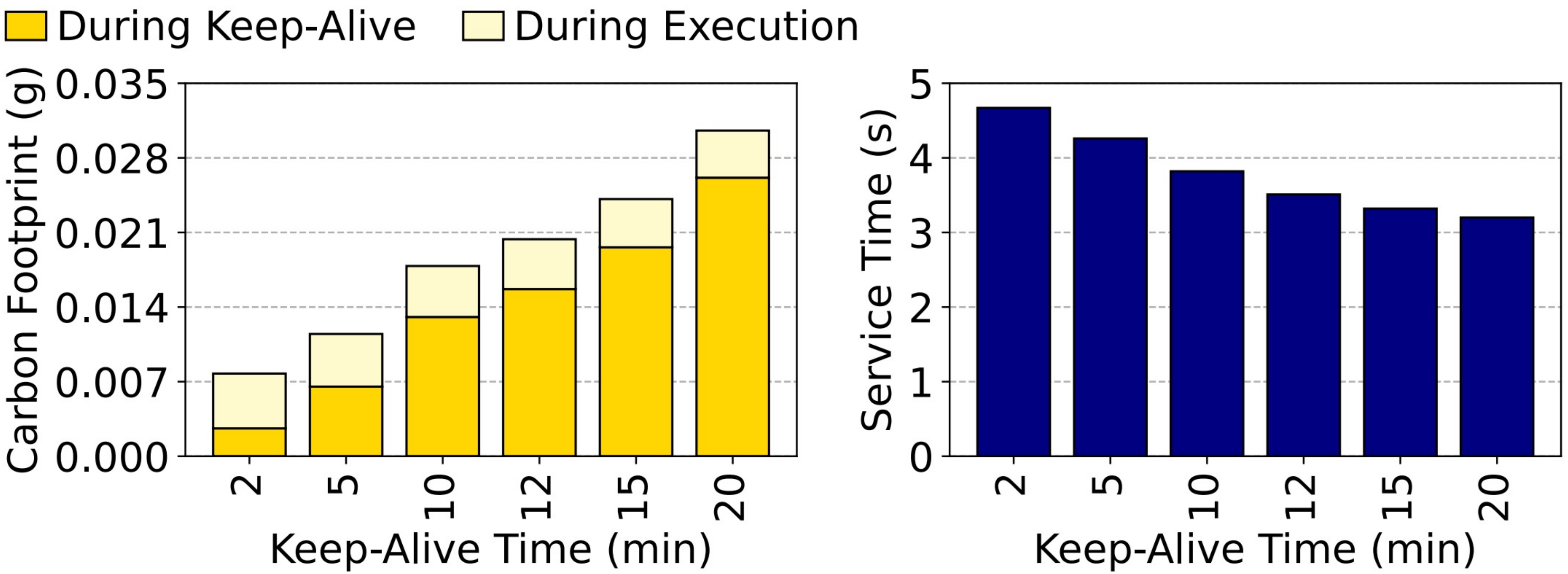
Carbon estimation varies with methodology; ignoring DRAM embodied carbon and CPU keep-alive underestimates serverless carbon footprints

Scheduling in heterogeneous hardware can bring opportunity to reduce the carbon footprint



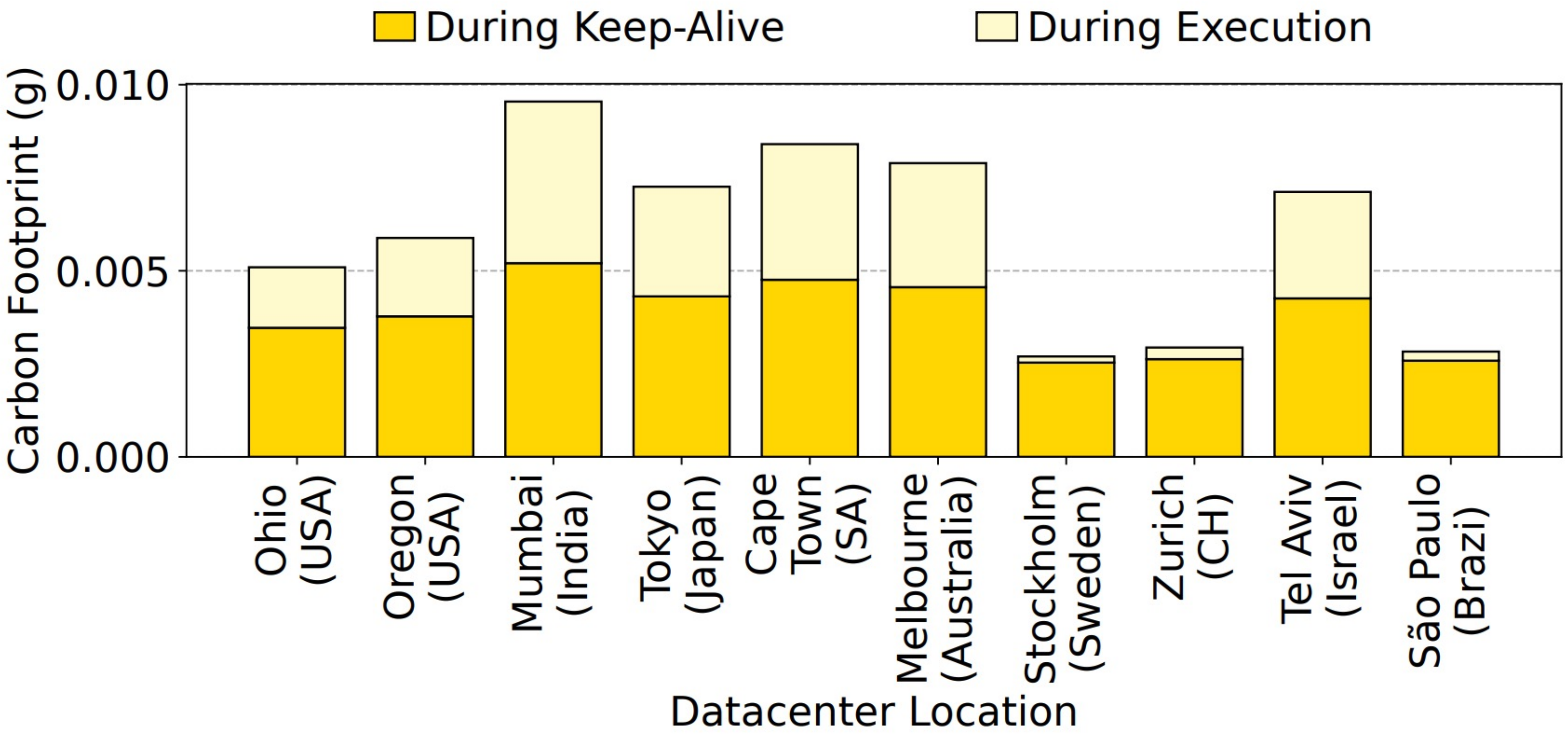
Carbon-optimal hardware varies by function and often differs from energy-optimal, shorter keep-alive durations suit newer hardware to reduce embodied carbon impact.

Performance and carbon footprint are at odds with respect to the keep-alive time of functions



Place high keep-alive functions on older hardware and high-energy functions on newer hardware

Carbon footprint varies across different datacenter Locations due to variations in carbon intensity



Use older hardware with lower embodied carbon for keep-alive in low-carbon regions, and energy-efficient hardware for execution in high-carbon regions

Carbon footprint accounting methodologies should be standardized in serverless computing

The Hidden Carbon Footprint of Serverless Computing

Rohan Basu Roy
Northeastern University

Raghavendra Kanakagiri
Indian Institute of Technology Tirupati

Yankai Jiang
Northeastern University

Devesh Tiwari
Northeastern University

ABSTRACT

Due to the unique aspects of serverless computing like keep-alive and co-location of functions, it is challenging to account for its carbon footprint. This is the first work to introduce the need for systematic methodologies for carbon accounting in the serverless environment, propose new methodologies and in-depth analysis, and highlight how the carbon footprint estimation can vary based on the chosen methodology. It discusses how serverless-specific scheduling choices can impact the trade-offs between performance and carbon footprint, with an aim toward standardizing methodological choices and identifying opportunities for future improvements.

CCS CONCEPTS

• **Computer systems organization** → Cloud computing

KEYWORDS

Serverless Computing, Carbon Footprint, Sustainability

ACM Reference Format:
Rohan Basu Roy, Raghavendra Kanakagiri, Yankai Jiang, and Devesh Tiwari. 2024. The Hidden Carbon Footprint of Serverless Computing. In *ACM Symposium on Cloud Computing (SoCC '24), November 20–22, 2024, Redmond, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3698038.3698546

1 Introduction And Background

Carbon emissions of computing systems. The greenhouse gas emissions from large-scale computing systems constitute more than 2.5% of global emissions [23, 29]. With the increasing demand for computing resources, the carbon footprint of computing is also increasing rapidly. Industries and cloud computing service providers are increasingly aiming to enhance the sustainability of their operations, focusing on reducing carbon emissions during the manufacturing of computing systems, energy consumption during operations, and using renewable energy during operations.

The carbon footprint of computing systems comprises the *operational* and *embodied* carbon. The *operational carbon* is the carbon emitted during the hardware operation, for the energy required for operation. It is a product of the energy consumption and the *carbon intensity* (CI). The CI is the amount of carbon emitted per unit of generated energy from a specific source (*i.e.*, if a power grid uses more renewable sources at a time, it delivers energy with lower CI). The CI varies temporally and spatially depending on the energy mix being used by a power grid of a location. The *embodied carbon* is the carbon emitted during the manufacturing of hardware, and it is amortized over the lifetime of hardware. Due to the effect of embodied carbon during manufacturing and varying levels of carbon intensity, minimizing carbon footprint is not the same as minimizing energy consumption, as also acknowledged by prior studies [21, 23] – although minimizing energy consumption is certainly helpful in also reducing carbon footprint.

Serverless computing and its carbon footprint. Serverless computing is a rapidly increasing form of cloud computing, with more than 70% of users of production cloud platforms like AWS, using one or more serverless solutions [11]. Since serverless functions are typically short-running, providers *keep alive* functions in the memory of servers when they are predicted to be invoked. If a function is not kept alive and it is invoked, it undergoes a *cold start*, otherwise, it undergoes a *warm start*. A higher keep-alive period increases the warm start chances. The summation of the execution time and the cold start time (0, if warm start) is the *service time* of a function.

While there have been attempts to measure and model the carbon footprint of applications running on a dedicated environment, they cannot be directly translated to estimate the carbon footprint of serverless computing. In this work, we argue that serverless computing has considerable environmental implications. While the stateless nature of serverless computing is attractive, the need to keep alive functions in memory incurs high embodied carbon emissions and should be accurately and robustly accounted for. Furthermore, the

SoCC '24, November 20–22, 2024, Redmond, WA, USA

ACM ISBN 979-8-4007-1286-9/24/11.
https://doi.org/10.1145/3698038.3698546

Recommendations for providers to perform carbon aware scheduling

Methodologies to measure carbon in the cloud

Insights on serverless carbon variation with application and hardware types

Future research should consider optimizing for both performance and sustainability, ensuring cross-platform consistency