# DUCT: Dynamic Unified Carbon Modeling Tool for Datacenter Scheduling

Wenkai Guan[1], Zerui Lyu[1], and Cristinel Ababei[2]
[1]Division of Science and Mathematics, University of Minnesota, Morris
[2]Department of Electrical and Computer Engineering, Marquette University
Email: {guan0210,lv000013}@morris.umn.edu, cristinel.ababei@marquette.edu

***Abstract*—Environmental sustainability in computing has drawn significant attention in the research community in recent years. There is an increasing demand-supply gap in computing, and the rise in computing demand incurs environmental overheads. To address the problem of increasing carbon emissions embodied into the manufacture and then use of computing systems, we present a dynamic unified carbon modeling tool (DUCT), which allows us to estimate at run-time (i.e., application scheduling) the embodied carbon emissions during datacenter operation. This in turn, allows to effectively include carbon emissions as an optimization objective, in addition to the traditional performance and energy efficiency objectives during job scheduling optimization. The proposed DUCT tool is integrated with a state-of-the-art scheduling algorithm. Simulation experiments on a real in-house heterogeneous computer cluster uncover significant differences in the actual carbon emission estimations by DUCT tool for several Splash-2 benchmarks, compared to when the estimations are made statically, at design time by existing tools. Therefore, the proposed DUCT tool can offer a more accurate run-time estimator of carbon emissions.**

***Index Terms*—Embodied carbon emissions; carbon modeling; datacenter; scheduling algorithm;**

## I. Introduction

Carbon emissions (i.e., $CO_2$) has become one of the most important challenges for computing systems. There is an increasing gap between the rate at which the performance of processors improves and the lower rate at which carbon emissions are reduced. For example, datacenters used an estimated 460 terawatt hours (TWh) of electricity in 2022, representing 2% of total global electricity demand, contributing around 3% to all global carbon emissions, and thereby exceeding emissions from commercial flights (about 2.4%) and other activities that fuel our global economy [1], [2]. Thus, the computing research community now faces the important challenge of reducing carbon emissions in all computing systems, but especially in datacenters that have increased in number and in types of applications that they run in the AI era, such as large power-hungry and computationally expensive AI models [3]. Therefore, we urgently need new approaches to designing and operating computing systems to optimize holistically carbon emissions. This paper addresses that need by proposing a Dynamic Unified Carbon modeling Tool (DUCT) to quantify carbon emissions during datacenter operation. It demonstrates an opportunity to significantly reduce embodied carbon emissions during datacenter job scheduling.

## II. Related Work and Motivation

### A. Related Work

In this section, we review previous literature on carbon modeling at the processor, cloud, and hierarchical (cross-layer) levels. In the category of processor-level carbon modeling approaches, most of the previous studies focused on the goal of sustainability at the design stage. The researchers in [4] proposed 3D-Carbon to quantify the carbon emissions of 3D and 2.5D integrated circuits during their life cycle. By considering the entire life-cycle of integrated circuits (ICs), the work in [5], [6] laid the foundation for holistic ICs level sustainability studies, which aim at identifying pathways towards truly green computing. The study in [7] analyzed the sustainability benefits of chiplet-based design choices by considering scaling, yields, design complexity, and advanced packaging techniques. The work in [8] proposed a carbon model under data uncertainty to assess processor sustainability.

In the category of cloud-level carbon modeling approaches, the study in [9] quantified carbon emissions of computer systems and found that most emissions related to datacenter equipment come from hardware manufacturing - that is commonly called *embodied carbon*. The work in [10] proposed a tool to balance the trade-off between embodied carbon and *operational carbon* for carbon-free datacenters. Their work is vital to reducing carbon emissions in datacenters at the design stage. The study in [11] introduced Google's system for global Carbon-Intelligent Computer Management to minimize carbon footprint and power cost. The study in [12] proposed carbon-aware approaches to reduce carbon emissions in datacenters. The work in [13] proposed to reuse unwanted smartphones as "junkyard computers" to reduce carbon emission due to new manufacturing. The study in [14], [15] co-optimized carbon emissions, water footprint, and energy cost of geo-distributed datacenters. The work in [16] proposed a carbon-aware scheduler to quantify the operational carbon, performance, and cost trade-off in the cloud. This work provides critical trade-off analysis, as it focuses on operational energy and carbon emissions.

In the category of hierarchical carbon modeling approaches, the work in [18] proposed ACT - an architectural carbon emission modeling framework to quantify embodied and operational carbon emissions at the design space exploration stage. The importance of quantifying and reducing the carbon

TABLE I
REVIEW OF CARBON MODELING AT THE PROCESSOR, CLOUD, AND HIERARCHICAL (CROSS-LAYER) LEVELS.

| Methods | Level | Stage | Embodied | Operational | Heterogeneity | Interference |
|---|---|---|---|---|---|---|
| [4] | processor | design | yes | yes | yes | |
| [5], [6] | processor | design | yes | yes | yes | |
| [7] | processor | design | yes | yes | yes | |
| [8] | processor | design | yes | yes | yes | |
| [10] | cloud | design | yes | yes | | |
| [11], [12], [16] | cloud | run-time | | yes | yes | |
| [13] | cloud | design | yes | yes | yes | yes |
| [14], [15] | cloud | run-time | | yes | yes | yes |
| [18] | hierarchical | design | yes | yes | yes | |
| **DUCT** | **hierarchical** | **run-time** | **yes** | **yes** | **yes** | **yes** |

footprint at every layer of the computing stack and across the life cycle of computing was again highlighted later in [17]. Our work in this paper is in this category. Building on these existing studies from [9], [4], [18], we further improve hierarchical carbon modeling by formalizing dynamic carbon modeling at run-time (*i.e.*, scheduling), by also considering heterogeneity and interference factors that affect application's execution time, and by investigating carbon emissions as an additional objective during job scheduling in datacenters.

### B. Motivation

The summary in Table I indicates that (*i*) we have excellent work at processor and cloud level approaches to quantify and reduce carbon emissions; (*ii*) we have only begun work on hierarchical carbon modeling approaches to quantify and reduce carbon across all layers of the computing stack, especially at run-time. Our work is in this space; it presents a hierarchical (server and datacenter) dynamic unified carbon modeling tool (*i.e.*, DUCT) that also considers heterogeneity and interference while modeling embodied and operational carbon at run-time. We are extending the basic framework introduced in [18] by applying it at the scheduling stage and using it to directly impact how job scheduling is done such that carbon emission is reduced. Furthermore, we adopt the carbon modeling idea from [18] and the heterogeneity and interference modeling ideas from [19], [20] and combine them with our new ideas on collaborative filtering based carbon prediction and D-Choices greedy scheduling [21], [22]. This paper is built on top of the previous work but differs from the previous one.

To motivate the need for DUCT, we show in Fig. 1 the carbon emission estimation done during the design stage by ACT and based on profiled execution and done during the scheduling stage by DUCT on the benchmark *barnes* from Splash-2 suite [23]. We can notice a significant difference between the two estimations, which emphasizes the need for a tool like DUCT to be used at run-time for more accurate estimations. To this end, the novelty of the proposed run-time carbon modeling framework lies in the following: 1) modeling of both embodied and operational carbon at the scheduling stage, 2) modeling also heterogeneity and interference in datacenters, and 3) applying these models during scheduling with the objective of reducing carbon emissions. The proposed unified carbon modeling represents a paradigm shift from the current datacenter scheduling work that focuses on optimizing energy efficiency toward embracing more sustainable methods in optimizing carbon emissions together with energy efficiency.

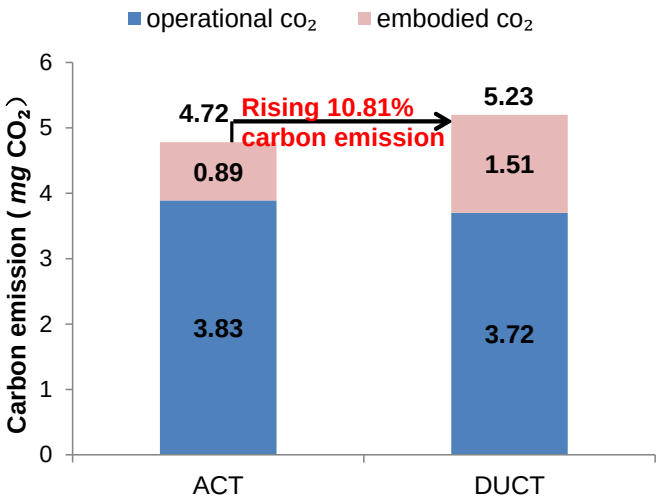


Fig. 1. Illustration of the difference in carbon emission estimation by ACT at design stage and by DUCT at runt-time scheduling stage for a representation benchmark, *barnes* from Splash-2 suite. The difference in application execution time at the scheduling stage from the profiled execution time during the design stage is the main reason for the gap in carbon emission estimation.

## III. DYNAMIC UNIFIED CARBON MODELING

The study in [18] proposed the ACT framework and laid the foundation for static (design time) unified carbon modeling. We develop our framework starting from ACT by applying it to capture the total carbon emissions - including embodied carbon and operational carbon - in the context of datacenter scheduling. We define scheduling as allocating applications to servers in datacenters and allocating tasks to cores in multicore processors on servers. Servers can be operated in different configurations defined by different Voltage/Frequency (V/F) levels, which are set dynamically by Dynamic Voltage and Frequency Scaling (DVFS) techniques.

The proposed dynamic carbon modeling tool, DUCT, captures both embodied and operational carbon as illustrated by the shaded boxes in Fig. 2. DUCT considers embodied carbon similarly to ACT, but, it is different from ACT in that it considers also carbon emissions in the run-time/scheduling stage (shaded blocks in Fig. 2). During scheduling optimization, the run-time carbon emissions are impacted by: 1) applications' execution time, which is affected by scheduling on different heterogeneous servers; this directly influences the run-time embodied carbon and 2) different server configurations defined by different voltage/frequency (V/F) settings as controlled dynamically by DVFS techniques; this directly influences the operational carbon.

In presenting details of the proposed dynamic unified carbon modeling framework, we will make use of various variables and parameters, which for clarity are summarized in Table II. We present our discussion in the context of a datacenter with a total number of $N$ servers, a number of $S_N$ server

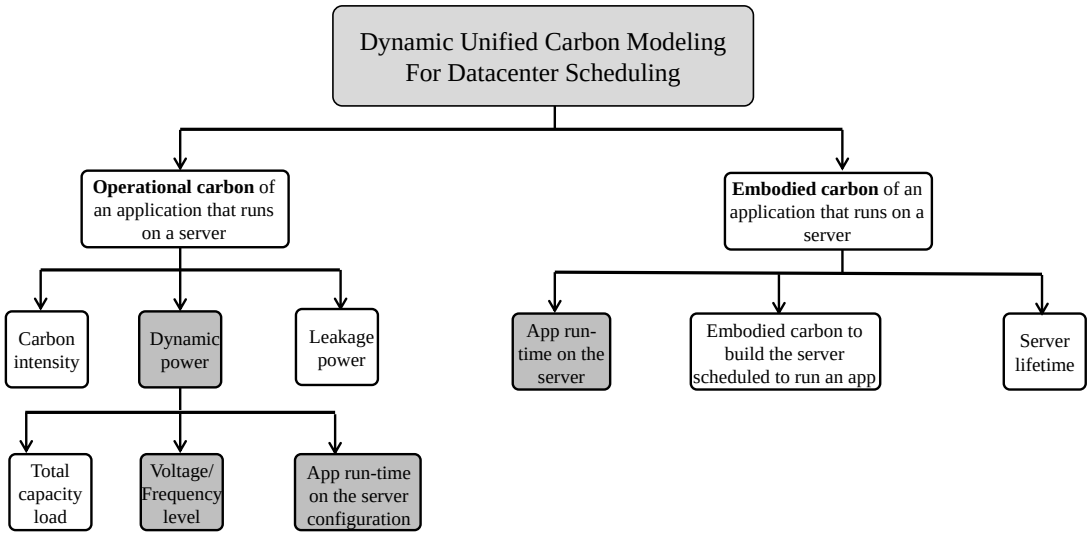


Fig. 2. Proposed dynamic unified carbon modeling tool. It accounts for both operational carbon and embodied carbon. It is build on top of ACT [18] but with the differences in (*i*) using application's execution time at the scheduling stage that maybe affected by the interference (contest in resources); this directly affects the run-time embodied carbon. (*ii*) using different server configurations defined by different voltage/frequency settings as controlled by DVFS techniques; this directly affects the operational carbon. We use shaded blocks to highlight these difference compared to ACT.

TABLE II
NOTATIONS AND DEFINITIONS USED IN THIS PAPER.

| Parameter | Description |
|---|---|
| $N$ | total number of servers |
| $S_N$ | total number of server configurations |
| $M$ | number of input applications |
| $S()$ | cluster and node levels scheduling functions |
| $C_{total}$ | total carbon emission due to scheduling $S()$ |
| $C(m,i)$ | carbon emission of application $m$ on server $i$ |
| $x_{m,i}$ | indicator, 1 if application $m$ runs on server $i$ and 0 otherwise |
| $m$ | an application |
| $E(m,i)$ | energy consumption of application $m$ on server $i$ |
| $CI_{use}$ | carbon intensity during use phase |
| $E(m,i)_{leakage}$ | leakage power |
| $E(m,i)_{dynamic}$ | dynamic power |
| $S_i$ | number of server configurations (e.g., V/F levels) for server $i$ |
| $E_{m,j}$ | energy usage of application $m$ on server configuration $j$ |
| $y_{m,j}$ | percentage of the execution time |
| $C$ | total capacitive load |
| $V$ | voltage, usually in-pair with frequency |
| $F$ | frequency, usually in-pair with voltage |
| $T_{m,j}$ | execution time of application $m$ on server configuration $j$ |
| $T(m,i)$ | application run-time on server $i$ |
| $LT(i)$ | server lifetime |
| $C(i)_{embodied,overall}$ | overall embodied carbon needed to build server $i$ |
| $C(i)_{embodied}$ | embodied carbon of different components in server $i$ |

configurations, and a number $M$ of input applications that need to be scheduled to these servers. Please note that a given physical server has several configurations, given by the total number of different V/F combinations supported on the given server. The scheduling algorithm that we use is based on the previous work [21], [22]. It is a unified scheduling approach that schedules applications to servers at the datacenter level and threads to cores at the node level; such a scheduling solution is denoted as $S()$ in this work. The total carbon emission $C_{total}$ of the scheduling function $S()$ is formulated as follows:

$$C_{total} = \sum_{m=1}^{M} \sum_{i=1}^{N} C(m,i) \cdot x_{m,i} \tag{1}$$

where $C(m,i)$ defines the carbon emission of application $m$ when it runs on server $i$. $x_{m,i}$ is an indicator variable that is 1 if application $m$ runs on server $i$ and 0 otherwise (we assume no cross-server migration in this paper). We consider both embodied carbon and operational carbon when estimating $C(m,i)$, using the following equation:

$$C(m,i) = C(m,i)_{embodied} + C(m,i)_{operational} \tag{2}$$

The operational carbon emission of an application $m$ when it runs on server $i$ is computed as the product between the energy consumed $E(m,i)$ and the carbon intensity during the use phase, $CI_{use}$. We adopt the concept of carbon intensity (to measure the equal grams of carbon dioxide per kWh of electricity) from the ACT framework [18], as it enables system developers to consider different energy sources (e.g., renewable energy with smaller $CI_{use}$ values or coal-based energy with larger $CI_{use}$ values). The expression used for calculating the operational carbon emission is thus:

$$C(m,i)_{operational} = E(m,i) \times CI_{use} \tag{3}$$

The energy usage $E(m,i)$ includes two main components that correspond to the dynamic and leakage power consumption components, which depend on the V/F settings at any particular time as dictated by the DVFS technique:

$$E(m,i) = E(m,i)_{dynamic} + E(m,i)_{leakage} \tag{4}$$

The dynamic component, $E(m,i)_{dynamic}$, is estimated with the following expression:

$$E(m,i)_{dynamic} = \sum_{j=1}^{S_i} E_{m,j} \cdot y_{m,j} \tag{5}$$

where $E_{m,j}$ defines the energy usage of application $m$ when it runs on server configuration $j$, out of $S_i$ total configurations for server $i$. $y_{m,j}$ represents the percentage of the execution

time (i.e., value in $[0, 1]$ range) when application $m$ runs on server configuration $j$ (we allow thread migration within a given server). The expression to calculate $E_{m,j}$ is:

$$E_{m,j} = P \times T_{m,j} \tag{6}$$

$$P = 1/2 \times C \times V^2 \times F \tag{7}$$

where $C$ is the total capacitive load, $V/F$ represent the voltage and frequency at which the server is configured, and $T_{m,j}$ is the execution time of application $m$ on the specific server configuration $j$.

The second main component in eq. (1) is the embodied carbon emission, $C(m,i)_{embodied}$, which is estimated as follows:

$$C(m,i)_{embodied} = C(i)_{embodied,overall} \times \frac{T(m,i)}{LT(i)} \tag{8}$$

where, similarly to the ACT framework [18], $C(m,i)_{embodied}$ uses the ratio between the application's execution time on server $i$ and the server lifetime $LT(i)$ (assumed fixed by the ACT framework). However, the key difference in our carbon model is that we use application's execution time at the scheduling stage, which maybe affected by the interference (contest in resources) based on different scheduling function $S()$, and thus influence the embodied carbon $C(m,i)_{embodied}$. For example, let's assume a scheduling function that allocates too many applications to a high-performance server, and some of the applications' execution time at the scheduling stage (focused by our carbon model) become longer than their execution time profiled individually at the design stage (focused by ACT) due to limited computational and memory resource. Under this circumstance, our carbon model can capture the changes in embodied carbon caused by scheduling functions and application's execution time at the scheduling stage.

In Eq. 8, $C(i)_{embodied,overall}$ represents the overall embodied carbon needed to build/manufacture that server. ACT framework quantifies it on a per-component level for: application processors (SoC), memory (DRAM), and storage (SSD and HDD) components. ACT does not consider PCB, battery, etc. due to lack of data currently. In this work, we adopt ACT framework's embodied carbon equations to quantify the overall embodied carbon $C(i)_{embodied,overall}$ for SoC, DRAM, SSD, and HDD.

## IV. Reduction of Carbon Emission During Scheduling Optimization

Previous sections answered the questions of why we need a dynamic unified carbon modeling tool and how to derive actual estimations based on such a model. This section aims to answer the question of how a dynamic unified carbon modeling tool benefits the datacenter scheduling. Based on the DUCT model presented in Section III, we here propose a hierarchical theoretical datacenter scheduling algorithm, which directly considers reduction of carbon emission as an optimization goal. This scheduling algorithm is developed by modifying a state-of-the-art scheduling algorithm that was presented in previous work by [21], [22]. The algorithm is a unified hierarchical cluster-node scheduling approach, which considers interference and server heterogeneity as a multi-objective optimization approach. It combines application-to-server (datacenter-level) and thread-to-core (server-level) scheduling problems, which are solved with novel D-choices greedy scheduling heuristics. In this paper, we extend that scheduling algorithm to include reduction of carbon emissions as an additional optimization objective. The proposed scheduling algorithm has a global view in that the server-level scheduling can influence the fast profiling results, which affect the datacenter-level scheduling. Meanwhile, the datacenter-level scheduling can enhance carbon reduction at the server-level through allocating applications to servers that have available lower embodied carbon.

**Algorithm 1:** Datacenter-level scheduling algorithm.

```
1  Inputs: Incoming applications M
2  Outputs: Application-to-server scheduling
3  Function DATACENTER-LEVEL-SCHEDULING()
4      for m in M do
5          BenchmarkProfiling(m); // Fast online profiling
6          CarbonModeling(m); //Dynamic unified carbon
             modeling
7          InterferenceModeling(m);
8          HeterogeneityModeling(m);
9          C = CarbonEmissionPrediction(m); //
             Collaborative filtering based carbon prediction
10         E = EnergyPrediction(m); // Collaborative
             filtering based energy prediction
11         I = InterferencePrediction(m); // Collaborative
             filtering based interference prediction
12         H = HeterogeneityPrediction(m); //
             Collaborative filtering based heterogeneity
             prediction
13         CarbonAwareScheduling(C, E, I, H);
14     end
15 end
```

Algorithm 1 listing describes the pseudocode of the datacenter-level scheduling algorithm. The inputs into this algorithm are the incoming applications, and the output is the application-to-server scheduling. Carbon emission is quantified by a new method $CarbonModeling$(m) during the optimization process. This method uses the models presented in Section III. We adopted the $BenchmarkProfiling$(m), $InterferenceModeling$(m), and $HeterogeneityModeling$(m) from [21], [22] to fast profile benchmarks' characteristics, which will be used to model interference and heterogeneity. In addition, we use collaborative filtering techniques to predict carbon emission (represented as $C$), energy usage (denoted as $E$), interference (represented as $I$), and heterogeneity (denoted as $H$) of new incoming applications. The predictions are used by $CarbonAwareScheduling$(C, E, I, H) to generate application-to-server scheduling that aims to reduce carbon emission along with energy consumption while guaranteeing

TABLE III
CHARACTERISTICS OF THE KUBERNETES CLUSTER.

| Server Type | Role | GHz | Cores | SSD (GB) | Mem(GB) |
|---|---|---|---|---|---|
| Intel i5-4590 | master | 3.30 | 4 | 5120 | 32 |
| Intel i7-4790 | worker | 3.60 | 8 | 128 | 16 |
| Intel i5-2500 | worker | 3.30 | 4 | 256 | 8 |
| Intel i5-2500 | worker | 3.30 | 4 | 512 | 8 |

TABLE IV
INPUT PARAMETERS IN DUCT CARBON MODEL.

| Parameters | Values | Descriptions |
|---|---|---|
| large_ssd | 2794 GB (3 TB) | main ssd |
| ssd | 2048 GB (2 TB) | secondary ssd |
| dram | 8 GB | memory size of each dram |
| carbon_intensity | 380 | 380g $CO_2$ per kWh in the USA |
| ssd_main_count | 1 | num. of main ssd |
| ssd_secondary_count | 1 | num. of secondary ssd |
| dram_count | 4 | num. of dram |
| cpu_count | 4 | num. of cores |

applications' performance needs. For more details on the main scheduling algorithm, please see [21], [22].

## V. EXPERIMENTAL RESULTS AND DISCUSSION

### A. Experimental Setup

In this section we first compare DUCT - the proposed tool for dynamic unified carbon modeling and estimation of carbon emission during datacenter operation - with ACT - the tool from [18] that estimated carbon emissions during the design stage. We conduct our experiments on a real in-house cluster built with four heterogeneous computers. Table III lists the specific characteristics of the in-house cluster testbed. Furthermore, we integrated the DUCT tool into the Qin Scheduler [21], [22], which is configured to optimize energy-delay-product and managed by the Kubernetes platform to distribute the applications to servers. We used Kubernetes v.1.14.0 and virtual networking layer flannel for communications. Because modern datacenters mainly include computational and latency-critical workloads, in our experiments, we use applications from the Splash-2 benchmark suite; more specifically: *barnes*, *ocean_cp*, *water_nsq*, *water_sp*, *lu_ncb*, and *radix* - which represent computational workloads. We keep most of the parameters inside the DUCT tool the same as ACT's default parameters corresponding to $28nm$ process technology node. However, we changed several parameters - specifically, those listed in Table IV - based on our particular cluster configurations. We used the DUCT parameters in the master computer as an example in Table IV.

### B. DUCT Captures Increase in Carbon Emission at Scheduling Stage

Here, we report the results from the first set of experiments: comparing carbon emissions estimated by ACT and DUCT tools. The results of our simulations are shown in Fig. 3, which plots the carbon emissions quantified by the previous work ACT tool (this is our base or reference for comparison) and by the proposed DUCT tool - for the investigated Splash-2 applications at the scheduling stage. The x-axis lists the Splash-2 applications, and the y-axis represents the carbon

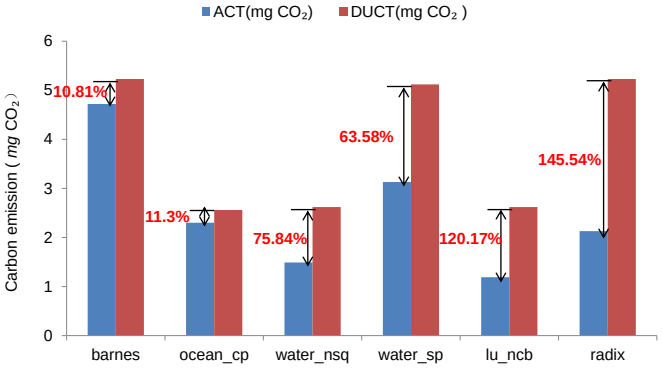


Fig. 3. Comparison of carbon emissions quantified by the proposed DUCT and previous ACT tools, for Splash-2 application benchmarks.

TABLE V
EMBODIED AND OPERATIONAL CARBON QUANTIFIED BY DUCT AND ACT FOR THE SPLASH-2 APPLICATIONS.

| | ACT ($mg\ CO_2$) | | | DUCT ($mg\ CO_2$) | | |
|---|---|---|---|---|---|---|
| App | Oper | Emb | Total | Oper | Emb | Total |
| barnes, | 3.83 | 0.89 | 4.72 | 3.72 | 1.51 | 5.23 |
| ocean_cp | 2.18 | 0.12 | 2.3 | 2.14 | 0.42 | 2.56 |
| water_nsq | 1.45 | 0.04 | 1.49 | 1.86 | 0.76 | 2.62 |
| water_sp | 2.92 | 0.21 | 3.13 | 4.29 | 0.83 | 5.12 |
| lu_ncb | 1.16 | 0.03 | 1.19 | 1.86 | 0.76 | 2.62 |
| radix | 1.76 | 0.37 | 2.13 | 3.72 | 1.51 | 5.23 |

emission in $mg\ CO_2$. We find that DUCT tool calculates an increase in carbon emission compared to ACT tool for all application benchmarks. To investigate whether the carbon increase is caused by embodied or operational carbon, Table V lists a break-down of the embodied and operational carbon emission for each of the studied applications. In this table, *Oper* represents operational carbon and *Emb* denotes the embodied carbon. It can be observed that embodied carbon dominates the increase in carbon emission during the scheduling stage.

This results can be explained as follows. First, ACT tool quantifies the application's carbon emission during the design stage and profiles application's execution time individually. Then, DUCT quantifies the application's carbon emission during the scheduling stage, at run-time, and uses the application's execution time that is affected by interference (e.g., application's contest due to limited computational and memory resources); this execution time is longer and results into the increased embodied carbon emission (according to Eq. 8). In our experiments, we noticed that for some applications, the operational carbon increased while for others it decreased. This could be explained by the fact that the operational carbon is affected by both the application's execution time (increased by interference) and by the different voltage/frequency pairs (selected by Qin Scheduler). This interaction between the impact of increased application execution time and selection of lower voltage/frequency level pairs is the focus of our ongoing investigation and it will be discussed in our future work.

### C. Optimization of Carbon-Performance-Energy Reduces Carbon Emissions with Little Impact on Energy Usage and Performance

In the second set of experiments, we implemented the datacenter-level scheduling from the listing Algorithm 1, which includes carbon emission as an additional optimization objective. This scheduling algorithm does what we call the *carbon-performance-energy* optimization scheduler. As a base or reference for comparison, we use the default scheduling algorithm from [21], [22], which focuses on optimizing the energy-delay-product - we call that the *performance-energy optimization* scheduler. We used the Kubernetes cluster described in Table III as our testbed. Similarly to [21], [22], as workloads, we used real-world applications from Splash-2 benchmark suite, randomly duplicated with equal likelihood to generate 10 real-world application workloads for our small cluster.

Our goal in these experiments is to compare the carbon-performance-energy optimization scheduler with the performance-energy optimization scheduler in order to investigate what is the impact of including carbon emission as an additional objective during scheduling. During our experiments, we collected job completion time, energy usage, and carbon emissions. We report in Table VI the summary of our results. We found that (*i*) carbon emission is reduced by 5.88%, but, at the expense of 3.91% increase in energy usage and at no impact on job completion time; (*ii*) the carbon-performance-energy optimization scheduler significantly reduced embodied carbon emissions (14.5%). This demonstrates that there exists an untapped opportunity to significantly reduce embodied carbon emissions at run-time via datacenter scheduling. Exploiting this opportunity represents a paradigm shift from optimizing energy efficiency alone to optimizing energy efficiency together with carbon emissions via more sustainable methods involving datacenter scheduling.

### D. Discussion

Based on the experimental results presented above, we note the following key differences between the previous work ACT and the proposed DUCT tools: 1) DUCT can capture application's execution time that is affected by interference during the scheduling stage; 2) DUCT can capture dynamic energy reduction due to dynamic voltage and frequency scaling (DVFS) for heterogeneous server configurations during the scheduling stage. The former affects the embodied carbon and both affect the operational carbon. The carbon-performance-energy optimization scheduler, which integrates the DUCT for carbon modeling, prefers scheduling jobs to servers with much lower carbon (e.g., older servers with much lower embodied carbon but still guarantee application's performance needs), but, at the cost of some increase in energy usage. The significant differences in carbon emissions that we have discovered in this paper highlight the need for tools like DUCT to be considered in run-time optimizations in datacenters. In essence, the proposed DUCT tool can complement the ACT tool in that it can provide dynamic unified carbon modeling employed at the scheduling stage in datacenters to enable the estimation and optimization of carbon emissions along with energy efficiency.

## VI. Conclusion

We presented a unified carbon emission modeling framework, which can be used as a tool to estimate dynamically the total embodied carbon emissions during datacenter operation. This framework allows us to directly include reduction of carbon emissions as an objective during datacenter optimization. More specifically, we employ the proposed framework to modify application scheduling in datacenters so that carbon emission is considered as an optimization objection in addition to the traditional power and performance objectives. Simulation experiments on using real-world applications on a custom in-house heterogeneous cluster showed that carbon emissions can be reduced while having little impact on energy usage and performance. This paper demonstrated a research opportunity to reduce carbon emissions through datacenter scheduling.


## Acknowledgement

We thank our anonymous reviewers for their constructive feedback. We appreciate the help from the EIT 2024 general co-chairs to make this paper possible.


## References


[1] IEA (2024), "Electricity 2024," *IEA*, 2024. Available: https://www.iea.org/reports/electricity-2024

[2] H. Lavi, "Measuring greenhouse gas emissions in data centres: the environmental impact of cloud computing," *Climatiq*, 2022.

[3] C.J. Wu et al., "Sustainable AI: Environmental Implications, Challenges and Opportunities," *MLSys*, 2022.

[4] Y. Zhao et al., "3D-Carbon: An Analytical Carbon Modeling Tool for 3D and 2.5D Integrated Circuits," *DAC*, 2024.

[5] E. Brunvand et al., "Dark Silicon Considered Harmful: A Case for Truly Green Computing," *IGSC*, 2018.

[6] D. K. Jr. et al., "GreenChip: A tool for evaluating holistic sustainability of modern computing systems," *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 322-332, June 2019.

[7] C. C. Sudarshan et al., "ECO-CHIP: Estimation of Carbon Footprint of Chiplet-based Architectures for Sustainable VLSI," *HPCA*, 2024.

[8] L. Eeckhout, "FOCAL: A First-Order Carbon Model to Assess Processor Sustainability," *ASPLOS*, 2024.

[9] U. Gupta et al., "Chasing Carbon: The Elusive Environmental Footprint of Computing," *HPCA*, 2021.

[10] B. Acun et al., "Carbon Explorer: A Holistic Framework for Designing Carbon Aware Datacenters," in *ASPLOS*, 2023.

[11] A. Radovanović et al., "Carbon-Aware Computing for Datacenters," *IEEE Transactions on Power Systems*, vol. 38, no. 1, 2023.

[12] L. Lin and A. A. Chien, "Adapting Datacenter Capacity for Greener Datacenters and Grid," in *e-Energy*, 2023.

[13] J. Switzer et al., "Junkyard Computing: Repurposing Discarded Smartphones to Minimize Carbon," in *ASPLOS*, 2023.

[14] S. Qi et al., "SHIELD: Sustainable Hybrid Evolutionary Learning Framework for Carbon, Wastewater, and Energy-Aware Data Center Management," in *IGSC*, 2023.

[15] S. Qi et al., "MOSAIC: A Multi-Objective Optimization Framework for Sustainable Datacenter Management," in *HiPC*, 2023.

[16] W. A. Hanafy et al., "Going Green for Less Green: Optimizing the Cost of Reducing Cloud Carbon Emissions," *ASPLOS*, 2024.

[17] T. Eilam, "Harnessing the Power of Specialization for Sustainable Computing," *ASPLOS*, 2024.

[18] U. Gupta et al., "ACT: designing sustainable computer systems with an architectural carbon modeling tool," in *ISCA*, 2022.

[19] F. Romero et al., "Mage: Online and Interference-Aware Scheduling for Multi-Scale Heterogeneous Systems," *PACT*, 2018.

TABLE VI
THE COMPARISON BETWEEN CARBON-PERFORMANCE-ENERGY OPTIMIZATION AND PERFORMANCE-ENERGY OPTIMIZATION FOR 10 REAL-WORLD APPLICATION WORKLOADS.

| | Carbon ($mg\ CO_2$) | | | Energy Usage ($Joule$) | Job Completion Time ($second$) |
|---|---|---|---|---|---|
| | Operational | Embodied | Total | | |
| Performance-Energy Optimization | 35.8 | 40.7 | 76.5 | 339.54 | 7 |
| Carbon-Performance-Energy Optimization | 37.2 | 34.8 | 72.0 | 352.25 | 7 |
| Difference | +3.91% | -14.5% | -5.88% | +3.74% | 0% |


[20] C. Delimitrou et al., "Paragon: QoS-aware scheduling for heterogeneous datacenters," *ASPLOS*, 2013.

[21] W. Guan and C. Ababei, "Qin: Unified Cross-layer Cluster-node Scheduling for Heterogeneous Datacenters," *IEEE Trans. on Sustainable Computing*, 2024.

[22] W. Guan and C. Ababei, "Unified cross-layer cluster-node scheduling for heterogeneous datacenters," *IGSC*, Oct. 24-27, 2022.

[23] S.C. Woo et al., "The SPLASH-2 programs: characterization and methodological considerations," *ISCA*, 1995.

[24] M.G. Moghaddam et al., "Dynamic Energy Management for Chip Multi-processors Under Performance Constraints," *Microprocessors and Microsystems*, 2017.

[25] J. Leverich et al., "Reconciling High Server Utilization and Sub-millisecond Quality-of-Service," *EuroSys*, 2014.