Towards Application Centric Carbon Emission Management

Sudarsun Kannan and Ulrich Kremer
Department of Computer Science, Rutgers University
New Brunswick, New Jersey, USA

ABSTRACT

Reducing the carbon emission of computing systems has become a first-order optimization goal distinct from optimizing for performance or energy consumption. Carbon emissions are due to application execution on a target system (operational emissions) and the production, transportation, and disposal of the system itself (embodied emissions). This paper investigates the impacts of different resource configurations in terms of available DRAM memory on the overall carbon emission of individual application executions. We first propose an application-centric carbon footprint model that considers DRAM and CPU. We then study the model using a widely-used key-value store (RocksDB) and Graph500 applications. The results for RocksDB indicate that the minimal emission configuration is application dependent and can lead to significant emission reductions compared to application-oblivious configurations that use higher DRAM capacity without improving performance. For some applications, small performance degradation may lead to substantial additional emission reductions.

CCS CONCEPTS

• **Software and its engineering → Memory management**.

KEYWORDS

Carbon footprint, DRAM, Operating Systems, Model

ACM Reference Format:
Sudarsun Kannan and Ulrich Kremer. 2023. Towards Application Centric Carbon Emission Management. In *2nd Workshop on Sustainable Computer Systems (HotCarbon '23), July 9, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604930.3605725

1 INTRODUCTION

Performance improvement, energy consumption reduction, and a reduced carbon footprint are distinct optimization goals. In this paper, we aim to illustrate and model the tradeoffs in this optimization space and provide guidelines to data center service providers as well as application users who want to reduce their carbon footprints.

In this paper, we focus on the performance, energy consumption, and carbon footprint of the compute (CPU) and memory system (DRAM main memory). The carbon footprint should reflect the carbon emission incurred during the manufacturing, transportation, and disposal of the computing system, i.e., its embodied emissions, as well as during its active operation, its operational emissions. Current literature reports that the majority of the carbon footprint of computing systems is attributed to its manufacturing process rather than its operation, and the footprint increases with hardware capabilities, such as more CPU cores, DRAM capacity, and others [7, 8, 15]. Modern CPUs have multiple cores, allowing an increase in performance by assigning more cores to an application, which also leads to an increase in energy consumption and associated operational carbon footprint. However, the higher performance also means that resources are freed up sooner and can be used by other application executions, thereby reducing the share of embodied emissions for each execution.

Different resource configurations have different impacts on the carbon footprint of an application or data center workload. For the purposes of this paper, a resource configuration specifies the amount of DRAM main memory available for each execution of an application or workload. The number of cores assigned to an application execution is assumed to be fixed to allow a better understanding of the impact of the memory size on the program performance and overall carbon emission, including the CPU.

Reducing carbon emissions in computing systems is a critical objective, and our focus lies specifically on minimizing the carbon footprint by reducing the DRAM capacity allocated to applications. Intriguingly, our empirical analysis reveals that when reducing the memory allocated to applications, the degradation in performance follows a non-linear pattern rather than a linear one. This finding suggests that optimal configurations of memory utilization exist for applications where memory reduction can be achieved without substantial performance degradation. Identifying and leveraging these optimal memory usage patterns can significantly reduce memory consumption and decrease carbon footprint without compromising overall system performance. To further support this, we propose a model that establishes a relationship between an application's memory usage and carbon footprint. This model considers an application's CPU utilization, memory accesses, and I/O characteristics to quantify its carbon emissions associated with different memory usage configurations. The model provides valuable insights into the potential reduction in carbon footprint achievable by optimizing memory utilization, thereby paving the way for designing more sustainable and efficient computing systems.

Experimental results based on a widely-used key-value store, RocksDB [1], and an in-memory graph application, Graph500 [13], show that the carbon emissions of the applications are non-linear with respect to the assigned memory sizes. Further, based on our experimental configuration and model, the embodied carbon footprint does not always dominate the operational carbon footprint.

The contributions of this paper are

- an application-centric view of carbon emissions that allows application-specific emission optimizations. Specifically, we show that by changing the amount of DRAM assigned to

HotCarbon '23, July 9, 2023, Boston, MA, USA

ACM ISBN 979-8-4007-0242-6/23/07…$15.00
https://doi.org/10.1145/3604930.3605725

individual application executions, carbon emissions can be optimized for different performance needs.
- the observation that different configurations with the same performance may have different carbon footprints (for example, RocksDB's [1] sequential data access).
- a better understanding of the tradeoffs between performance and carbon emission that users or data center service providers can use to establish different levels of environmentally friendly, low carbon emission application execution modes.

Our future work will include a more detailed evaluation of the configuration spaces (e.g., number of CPU cores, GPUs, SSDs, HDDs) and their carbon emission models across a wide range of applications. It is also still an open problem how the observed application behaviors under different configurations can be effectively exploited through offline and online strategies, i.e., how the best configurations can be selected for each application execution in a multi-tenant environment.

2 BACKGROUND

In recent years, reducing the carbon footprint in datacenter systems has become a significant research focus due to the escalating environmental concerns and the increasing energy demands of modern computing infrastructures. A key aspect of this research is the development of models and strategies that effectively mitigate the environmental impact of datacenters while maintaining their operational efficiency.

2.1 Studies on Carbon Footprint Reduction

Numerous studies have explored strategies and techniques to reduce the carbon footprint in datacenter systems. This subsection provides an overview of some key research directions in this domain.

Hardware Design and Architectural Innovations: Studies have explored hardware-level improvements and architectural innovations to enhance the energy efficiency of datacenter systems [3]. This includes advancements in server design, compute, memory, and storage subsystems, and power delivery mechanisms. As prior studies have shown, hardware optimizations such as low-power processors, energy-aware memory systems (e.g., LPDDR [7]), and efficient power distribution contribute to reducing the carbon footprint of datacenters.

Energy-Efficient Resource Allocation: One prominent research direction involves optimizing resource allocation in datacenters to minimize energy consumption [9, 14]. For example, studies have proposed efficient dynamic resource provisioning algorithms and workload consolidation techniques to allocate resources based on demand. These approaches consolidate workloads onto fewer servers, thereby reducing power consumption and lowering carbon emissions.

Renewable Energy Integration: Another area of focus revolves around integrating renewable energy sources into datacenter operations. Researchers have explored the utilization of solar, wind, and hydroelectric power to supplement or replace traditional power sources [10]. Integrating renewable energy into datacenters helps reduce reliance on fossil fuels, significantly reducing carbon emissions. A recent U.S. EPA report shows a high 0.433 kg CO2 per kWh emission. As we show, these emissions can significantly increase both operational and embodied carbon footprints [4].

Power Management Techniques: Besides the above approaches, various power management techniques have been investigated to optimize energy usage within datacenters. These techniques range from CPU throttling to dynamic voltage frequency scaling (DVFS) and workload scheduling algorithms. By managing power consumption at the server and the datacenter levels, these approaches aim to improve overall energy efficiency [6].

2.2 Model for Carbon Footprint Reduction:

One crucial aspect of reducing the carbon footprint in datacenters is the development of models that quantify the relationship between energy consumption and carbon emissions. Such models provide a basis for evaluating the carbon emission impact of datacenter operations and identifying improvement opportunities.

More recently, researchers proposed an Architectural Carbon Modeling Tool (ACT) [7]. ACT facilitates carbon-aware design space exploration in hardware design. ACT consists of an analytical, architectural carbon footprint model and optimization metrics tailored to specific use cases. The tool enables the estimation of the carbon footprint of hardware by considering factors such as workload characteristics, hardware specifications, semiconductor fab characteristics, and environmental factors.

One of the critical contributions of ACT is the inclusion of embodied carbon in the carbon model. Embodied carbon refers to the emissions generated during the design and manufacturing phase of hardware, which are then spread over the lifetime of the hardware platform. The model combines operational carbons, the carbon emissions resulting from energy consumption during application execution, and embodied carbon, which includes packaging overheads and the embodied carbon of individual hardware components.

ACT proposes four sustainability-driven optimization metrics to aid in sustainable system design exploration. These metrics enable balancing carbon emissions with other performance and energy considerations specific to different use cases. Examples of these metrics include carbon-delay-product for balancing carbon and performance in data center scenarios and carbon-energy-product for balancing carbon and energy in sustainable mobile devices. The choice of optimization metric depends on the dominant factor contributing to the overall carbon footprint.

Overall, ACT addresses the need for quantifying and incorporating sustainability in hardware design space exploration.

2.3 Open Challenges

While significant progress has been made in reducing carbon emissions in datacenter systems, challenges remain. Factors such as workload diversity and hardware resource heterogeneity (e.g., memory and compute heterogeneity), scalability, and cost considerations pose ongoing research questions. Addressing these challenges requires holistic approaches that combine innovative modeling, system-level optimizations, and efficient resource management strategies. While in this work, we focus on application-centric modeling with respect to memory and compute, a wider analysis of system resource use (e.g., GPU) with respect to carbon footprint is required, both for single application executions as well as multiple application executions in a multi-tenant environment.

3 NEED FOR APPLICATION-CENTRIC CARBON EMISSION MODEL

Recent work has shown that carbon emissions due to production, transportation, and disposal, i.e., *embodied carbon emissions* (E_{CF}) of a computing system can be significantly larger than its *operational carbon emissions* (OP_{CF}) defined as the carbon emissions needed to produce the energy to operate the system [7, 8, 15]. A basic model for the carbon emissions of an application, CE_{app} running on a computing system is therefore

$$CE_{app} = t_{app} * OP_{CF} + \frac{t_{app}}{lifetime} * E_{CF}$$

where t_{app} is the execution time of the application and $lifetime$ is the lifetime of the computing system, typically between two and five years. In our evaluation, we use three years. This basic model does not consider the use of renewal energy during manufacturing or operation of the computing system which can significantly impact the overall carbon emissions (carbon intensity). We will need to refine this basic model to account for different resources available for an application's execution, and the resulting impact on its performance, operational emissions, and embodied emissions. A resource profile, called a *configuration* ($conf$), is a vector of compute, memory, and storage resources with their specific amounts. For the purposes of this paper, we limit a configuration to the amount of available main memory (DRAM). The overall carbon emission that can be attributed to an application's execution under a given configuration is modeled as:

$$CE_{app}(CPU) = t_{app}(CPU) * OP_{CF}(CPU) + \\ \frac{t_{app}(CPU)}{lifetime} * E_{CF}(CPU)$$

$$CE_{app}(DRAM) = t_{app}(DRAM) * OP_{CF}(DRAM) + \\ \frac{t_{app}(DRAM)}{lifetime} * E_{CF}(DRAM)$$

$$CE_{app}(conf) = CE_{app}(conf.CPU) + CE_{app}(conf.DRAM)$$

The key difference to previous work is that each application is "charged" for the amount of embodied and operational carbon emissions and types of resources it is using, i.e., the configuration for its execution. In other words, our approach is application focused rather than supporting, for example, architectural system design explorations [7]. The configuration space for an application can be rather large and may have different configurations for best performance, energy usage, and carbon footprint. A configuration space may also be limited due to shared resources in a multi-tenant environment. If performance is the main consideration, we are interested in finding the best configuration $conf\_min$ that minimizes the carbon footprint of an application while respecting a given performance constraint.

Find $conf\_min \in CONF\_SPACE$ such that

(1) $CE_{app}(conf\_min)$ is minimal and
(2) $t_{app}(conf\_min) \leq (1 + tolerance) * t_{app}(conf\_fastest)$

where $conf\_fastest$ is the configuration with the best performance, and $tolerance \geq 0$ is the performance reduction that is acceptable to the application user to achieve a better carbon emission outcome. For example, an acceptable performance reduction of 10% is specified as $tolerance = 0.1$.

CPU Parameters	
Parameter	Value
Power Consumption per Core	0.01 kW
US Energy to Carbon Conversion	0.433 kg CO2/kWh [4]
Core Area	1 cm^2
Number of Cores	8
Energy per Area	0.90 kWh/cm^2 [7]
Gas per Area	0.175 kg CO2/cm^2 [7]
DRAM Parameters	
Parameter	Value
Active Power (8 GB)	0.003 kW
Carbon Footprint per GB	0.6 kg CO2/GB [7]

Table 1: Hardware Parameters and Constants Used.

4 EXPERIMENTAL RESULTS

We first describe our approach to estimating carbon footprint and our overall assumptions, followed by our analysis of two widely studied applications.

Algorithm 1 Calculate CPU and DRAM Carbon Footprint

```
cpu_rt_lifetime ⟵ app_runtime / cpu_lifetime
dram_rt_lifetime ⟵ app_runtime / dram_lifetime

cpu_energy ⟵ app_runtime × power_per_core × num_cores
energy_embodied ⟵ energy_per_area × US_energy_to_carbon
cpu_carbon_footprint ⟵ (gas_per_area + energy_embodied) × core_area × num_cores × cpu_rt_lifetime

dram_energy ⟵ (memory_size_GB / 8) × DRAM_power_per_8GB × app_runtime
dram_carbon_embodied ⟵ memory_size_GB × DRAM_carbon_footprint_per_GB × dram_rt_lifetime
dram_carbon_active ⟵ dram_energy × US_energy_to_carbon
dram_carbon_footprint ⟵ dram_carbon_active + dram_carbon_embodied
```

4.1 Approach to Estimating Carbon Footprint

In this study, we focus specifically on analyzing the carbon footprint of CPU and DRAM while deferring a comprehensive examination of end-to-end hardware, including storage and network, for our future research. A fundamental aspect of our approach is the assignment of operational and embodied carbon footprints to individual applications. Algorithm 1 outlines the methodology employed to calculate the CPU and DRAM carbon footprint, taking into account various hardware and carbon footprint parameters as depicted in Table 1. The algorithm incorporates factors such as runtime, power consumption, energy per area, gas emissions per area, core area, and the number of cores in its calculations.

For our analysis, we use the hardware parameters identified using prior studies, as shown in Table 1. For example, for the power consumption per core, we use 0.010 kWh, area per core is 1 cm^2, and the number of physical cores is 8 (with 32 threads). For CPU and DRAM embodied value parameters, we use the values from prior published studies [7] and other available resources. These values serve as estimates that allow the assessment of the impact of a reduced DRAM allocation on the performance and carbon footprints of CPU and DRAM.

First, we calculate the active energy consumption for each memory capacity by multiplying the runtime by the power consumption per core. Then, we calculate the carbon emissions by multiplying the active energy consumption by the energy to the carbon conversion factor, using the recently published report from the Environmental Protection Agency [4].

Next, we calculate the embodied carbon footprint per core by multiplying the gas emissions per area and energy per area with the area per core. This represents the carbon footprint associated with the physical manufacturing of the CPU. We use the values identified by a recent study [7] for our study. We also calculate the total area occupied by the CPU cores by multiplying the area per core by the number of cores. To calculate the total carbon footprint for a given memory capacity configuration, we combine the operational carbon emissions and the embodied carbon footprint.

The algorithm also measures the carbon footprint of DRAM. For our particular experiments, we observe that the carbon footprint of DRAM is lower than the carbon footprint of our selected 8-core CPU.

4.2 Application Analysis and Hardware Setup

To understand the performance vs. memory capacity tradeoff and their resulting carbon footprint implications, we select a popular key-value store, RocksDB, and the popular graph500 benchmark.

RocksDB [1] is a widely used, high-performance key-value store that offers data reliability and durability. It is commonly used as a backend for large-scale applications, including machine learning workflows. RocksDB uses a log-structured merge tree, which efficiently utilizes both memory and disk space. It leverages a configurable in-memory caching mechanism to enhance read performance by minimizing disk I/O operations. Therefore, RocksDB directly benefits from higher memory capacity. Additionally, RocksDB implements memory management techniques to strike a balance between performance and memory usage. Previous research has demonstrated that, in real-world scenarios, the memory capacity is often much smaller than the actual size of the database, and the application's access patterns can exhibit significant variations.

Graph500: Graph500 [13] is a benchmark designed to assess the performance of computing systems when solving graph-based problems. It specifically focuses on evaluating the memory and computational capabilities of these systems. The benchmark evaluates the efficiency of graph algorithms in terms of memory utilization, which includes both CPU and DRAM. We utilize the breadth-first search (BFS) and Single-Source-Shortest-Path (SSSP) benchmarks as representative workloads. Unlike RocksDB, this application does not involve I/O operations. However, insufficient memory capacity can lead to disk swapping and significantly impact performance.

Hardware Setup: For our analysis, we use the CloudLab system, with two memory sockets totaling 120GB, a 64-core, 2.8 GHz AMD 7543 processor, and a 1.6 TB NVMe SSD. The NVMe SSD offers maximum read and write bandwidths of 1.4 GB/s and 0.9 GB/s, respectively. We run all applications on a Linux 5.14 kernel.

4.3 Impact of Memory + I/O Heavy RocksDB

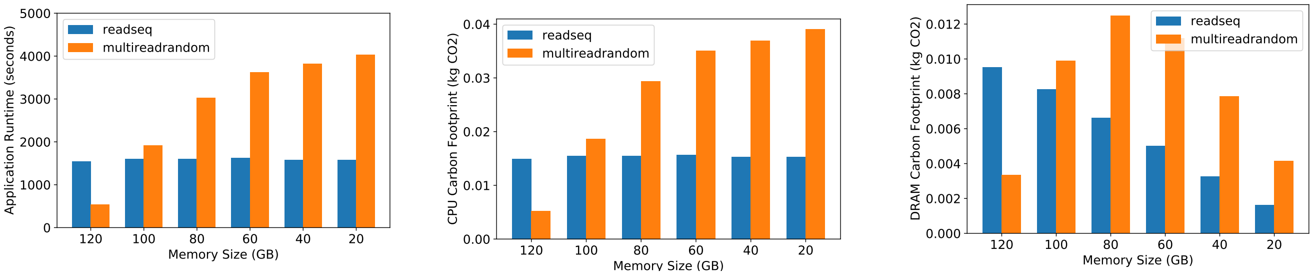
We analyze our model with RocksDB, which is both memory and I/O intensive. This study mainly focuses on the correlated impact of memory capacity and the corresponding impact of CPU and memory carbon footprint.

For our analysis, we vary the memory capacity sizes from 20GB to 120GB, representing RocksDB deployment under different amounts of memory on a system. For RocksDB, which generates a NoSQL database, we use a database size of 120GB, which corresponds to 40 million key-value pairs and a 4KB value size. It is worth noting that the case where the database size is equal to the available memory size is idealistic and impractical in real-world scenarios where the application's memory size is often provisioned with far lower capacity than the total size of the database.

We evaluate two common access patterns, namely random read ("multi-readrandom") that fetches more than one key for each access and sequential read ("readseq") that performs pointed queries. The purpose of the experiment is twofold: first, to examine the effect of memory size on throughput for different access patterns, and second, to analyze the corresponding carbon footprint. To conduct our analysis, we employ the commonly used *dbbench* workload [2]. To maintain brevity, we present only the overall runtime required to complete the workload. However, it is important to note that the overall application throughput is directly proportional to the runtime for *dbbench.*

RocksDB Memory Capacity vs. Performance Tradeoffs: Figure 1a shows the performance (i.e., runtime) vs. memory size implications. The results reveal interesting variations in performance based on memory size and access pattern. For the random access pattern, higher memory sizes lead to significantly improved speedups. We observe the highest throughput (and lowest runtime) of approximately 830K ops/sec with a memory size of 120GB. However, it is important to note that the degradation is not linear for the random read workload beyond the idealistic case of using the entire memory for the database. The throughput decreases gradually as the memory size decreases, but the degradation rate slows down. This suggests that there may be diminishing returns in terms of performance improvement as memory sizes continue to decrease. It also means that for smaller memory sizes, the application can tolerate significant reductions in available memory (e.g.: from 60 GB to 20 GB) with only little impact on performance.

On the other hand, the sequential read access pattern exhibits a different trend. The throughput remains relatively stable across different memory sizes, ranging from around 400K to 500K ops/sec. This is because, even with a low memory budget, for sequential access, the OS continuously prefetches (next N) sequential blocks from the disk to memory, which avoids the need to load the entire database into the memory, also reducing the active working set size. This suggests that the sequential read access pattern is less sensitive to changes in memory size than read-random, i.e., the

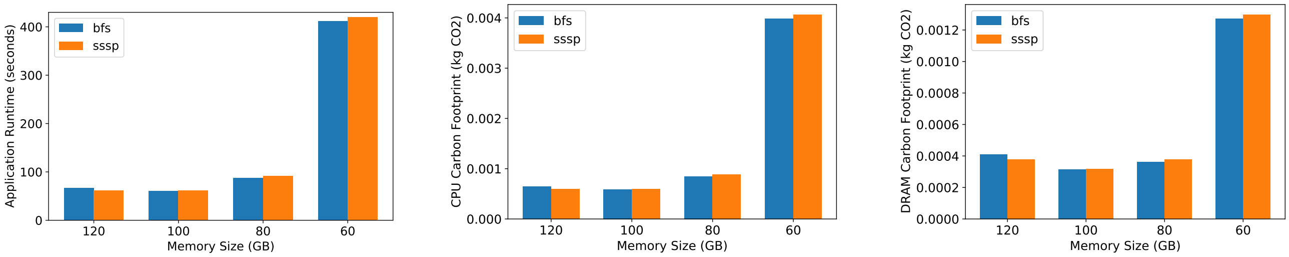


(a) Runtime

(b) CPU Carbon Footprint

(c) DRAM Carbon Footprint

Figure 1: RocksDB: Carbon Footprint and Runtime Analysis



(a) Runtime

(b) CPU Carbon Footprint

(c) DRAM Carbon Footprint

Figure 2: Graph500: Carbon Footprint and Runtime Analysis

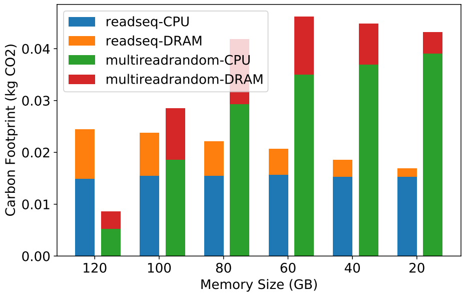


Figure 3: RocksDB Total Carbon Footprint

application can tolerate significant memory reductions without affecting performance. Note that, as prior studies have highlighted, sequential access patterns are quite common on several large-scale systems and for real-world workload traces [5].

Memory Capacity vs. CPU Footprint: Figure 1b and Figure 1c compare the carbon footprint of CPU and DRAM for a single application run and Figure 3 shows the total CPU + DRAM carbon footprint. To identify the embodied and active carbon footprint, we use the values from prior work [7].

First, for the random read access pattern, with a hypothetical scenario where the dataset fully fits in memory (120 GB). In this case, the application runtime is significantly shorter, leading to a substantial reduction in the CPU carbon footprint. However, as we reduce the memory size, the application incurs I/O operations, resulting in longer runtimes and an overall increase in the CPU carbon footprint.

However, when the memory capacity is half (60 GB) or lower than half the database size, the access overheads shift to I/O. As a result, further reducing the memory capacity along the x-axis shows that the runtime does not change significantly. For instance, when using only 20 GB of the memory capacity, compared to the 60 GB case, the CPU carbon footprint increases by only 11%. This reduction in memory capacity can effectively reduce both embodied and operational carbon footprints.

Memory Capacity vs. DRAM Footprint: The carbon footprint of DRAM is primarily influenced by the memory configuration utilized and its impact on runtime. This relationship is depicted in Figure 1c, showcasing interesting trade-offs between memory capacity and carbon footprint.

When employing a higher memory capacity (e.g., 120 GB), where the entire application dataset fits in memory (a hypothetical setup), the application runtime is significantly shorter, which dominates in reducing the overall DRAM carbon footprint despite increasing the carbon footprint from higher capacity. Conversely, as the DRAM capacity is reduced, the runtime substantially increases, leading to a higher overall carbon footprint. However, beyond a certain threshold (e.g., 60 GB), the advantages of lower DRAM capacity from reduced operational and embodied costs outweigh the increase in runtime, once again leading to a reduction in the overall carbon footprint.

These results show that by carefully managing memory resources and selecting an appropriate memory size based on workload requirements, substantial reductions in the overall carbon footprint can be achieved, promoting sustainability in computational operations.

4.4 Memory-intensive Graph500

We next evaluate Graph500 using BFS and SSSP benchmarks that use 32 application threads on 8 physical cores. In Figure 2, we show the runtime and the impact on CPU and DRAM carbon footprint. The evaluation of the Graph500 benchmark provides valuable insights into performance, memory capacity, and carbon footprint.

Firstly, in our Graph500 workload, approximately 99 GB of memory is required. As a result, there is no significant difference in performance between using 120 GB and 100 GB. However, accurately identifying and utilizing the actual memory required by the application can help reduce both CPU and DRAM carbon footprints.

Furthermore, we observe that the runtime of the Graph500 benchmark tends to increase as the memory size decreases. This suggests that larger memory sizes facilitate faster execution of graph algorithms. For instance, when the memory size is reduced to 80 GB, the runtime increases by 1.35×, consequently increasing the CPU carbon footprint by 1.3×. Subsequently, further reducing the memory to 60 GB leads to a runtime increase of 5.1×, mainly due to the need to swap memory to disk, which generally incurs slower performance. Consequently, both the DRAM and CPU footprints also increase. Graph500's carbon emissions exhibit a strong stepwise behavior where crossing a minimal memory threshold leads to a significant increase in the overall carbon footprint. These findings demonstrate that, unlike in RocksDB's sequential read access pattern, reducing memory capacity for certain types of applications could substantially increase the overall runtime and CPU carbon footprint. Identifying this memory threshold will be crucial for an effective carbon emission management.

5 CARBON EMISSION MANAGEMENT STRATEGIES

We envision different carbon emission strategies that either use models to determine a configuration that represents the desired tradeoffs between performance and carbon emission or use dynamic, feedback-directed algorithms to converge on such a configuration. These strategies can be implemented within the operating system or through specialized runtime system support.

Previous work has used machine learning strategies to build off-line performance and energy models for applications and their configuration spaces [11, 12]. We will build regression models that map configurations, CPU footprints, and memory access patterns to performance, energy consumption and carbon emission. The CPU footprints and memory access patterns are measured through performance counters. During the training phase of an applications, different configurations with different workloads are used to collect the data needed for training. As in [11], we will experiment with different regression and machine learning approaches to determine the best model for each application. Different CPU footprints and memory access patterns may be clustered in order to simplify the models.

The minimal configuration with acceptable performance (within specified tolerance) is computed at runtime as a solution to a constrained integer optimization problem or as an approximation to such a solution. Reconfigurations may be necessary in a multi-tenant system due to sharing of common resources or changes in data access patterns within a single application. The runtime costs of online observation and reconfiguration overheads have to be taken into account. However, we believe that such costs are small compared to the expected benefits of configuration optimization for carbon emission reduction.

Extending OS Memory Management: Because the OS is responsible for per-application and system-wide memory allocation, we aim to design a feedback-driven OS-level memory management extension to identify the minimal memory required by an application without increasing CPU runtime, thereby keeping the memory carbon footprint low. The OS memory manager plays a critical role in managing the allocation and deallocation of memory resources for running applications. To optimize memory usage and reduce the carbon footprint and environmental impact, the memory manager can employ various techniques.

Our approach is to leverage sophisticated memory profiling and analysis algorithms. By analyzing the application's memory access patterns and resource utilization during runtime, the memory manager can gain insights into the application's memory requirements. As shown in our experimental results on RocksDB, for applications performing sequential data access, the memory footprint could be significantly reduced by identifying the portions of memory that are frequently accessed and those that are rarely used and moving the data in and out of memory without substantial carbon overhead. We also expect changes to the OS scheduling mechanisms with a focus on reducing the carbon footprint.

6 CONCLUSION AND FUTURE WORK

Improving the carbon emissions of individual application executions in addition to entire computing systems has become a first-order optimization objective. Our presented results indicate the potential benefit of carbon-aware resource assignments for applications and data access patterns such that their carbon footprint is minimized with no or only slight performance penalties.

Our future work will also include the investigation of an extended configuration space beyond DRAM, including heterogeneous memory (e.g.: SSD, HDD) and heterogeneous computing architectures (e.g.: GPU, ASIC, FPGA). Applications may have a limited set of data access patterns that may be included as part of a configuration (input dependencies). Further, there is a need for new optimization strategies that require novel offline and online approaches, which we will further investigate in the future. Finally, optimizing the carbon emissions of multi-tenant systems will be an important future challenge.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful comments and feedback. We would also like to acknowledge the CloudLab infrastructure support (NSF #1419199) for providing the necessary resources to conduct these experiments. Finally, Sudarsun Kannan was supported by funding from NSF CNS #1910593.

REFERENCES

[1] [n. d.]. RocksDB. http://rocksdb.org/.

[2] [n. d.]. RocksDB DBBench. https://github.com/facebook/rocksdb/wiki/Benchmarkingtools/

[3] Noman Bashir, Tian Guo, Mohammad Hajiesmaili, David Irwin, Prashant Shenoy, Ramesh Sitaraman, Abel Souza, and Adam Wierman. 2021. Enabling Sustainable Clouds: The Case for Virtualizing the Energy System. In *Proceedings of the ACM Symposium on Cloud Computing* (Seattle, WA, USA) *(SoCC '21)*. Association for Computing Machinery, New York, NY, USA, 350–358. https://doi.org/10.1145/3472883.3487009

[4] Environmental Protection Agency. 2023. Greenhouse Gas Equivalencies Calculator. https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator

[5] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '12)*. London, England, UK.

[6] M. Garcia Bardon, P. Wuytens, L.-Å. Ragnarsson, G. Mirabelli, D. Jang, G. Willems, A. Mallik, A. Spessot, J. Ryckaert, and B. Parvais. 2020. DTCO including Sustainability: Power-Performance-Area-Cost-Environmental score (PPACE) Analysis for Logic Technologies. In *2020 IEEE International Electron Devices Meeting (IEDM)*. 41.4.1–41.4.4. https://doi.org/10.1109/IEDM13553.2020.9372004

[7] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 784–799. https://doi.org/10.1145/3470496.3527408

[8] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2020. Chasing Carbon: The Elusive Environmental Footprint of Computing. *CoRR* abs/2011.02839 (2020). arXiv:2011.02839 https://arxiv.org/abs/2011.02839

[9] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *J. Mach. Learn. Res.* 21, 1, Article 248 (jan 2020), 43 pages.

[10] Kewen Li, Huiyuan Bian, Changwei Liu, Danfeng Zhang, and Yanan Yang. 2015. Comparison of geothermal with solar and wind power generation systems. *Renewable & Sustainable Energy Reviews* 42 (2015), 1464–1474.

[11] Liu Liu, Sibren Isaacman, and Ulrich Kremer. 2020. Global Cost/Quality Management across Multiple Applications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE'20)*. Association for Computing Machinery, New York, NY, USA, 350–361. https://doi.org/10.1145/3368089.3409721

[12] Liu Liu, Sibren Isaacman, and Ulrich Kremer. 2022. An Adaptive Application Framework with Customizable Quality Metrics. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 27, 2 (2022), 1 – 33. https://doi.org/10.1145/3477428

[13] Masahiro Nakao, Koji Ueno, Katsuki Fujisawa, Yuetsu Kodama, and Mitsuhisa Sato. 2021. Performance of the Supercomputer Fugaku for Breadth-First Search in Graph500 Benchmark. In *High Performance Computing: 36th International Conference, ISC High Performance 2021, Virtual Event, June 24 – July 2, 2021, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 372–390. https://doi.org/10.1007/978-3-030-78713-4_20

[14] Satveer and Mahendra Singh Aswal. 2016. A comparative study of resource allocation strategies for a green cloud. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. 621–625. https://doi.org/10.1109/NGCT.2016.7877487

[15] Swamit Tannu and Prashant J. Nair. 2022. The Dirty Secret of SSDs: Embodied Carbon. arXiv:2207.10793 [cs.AR]