In the next decade, datacenters will account for over 20% of global carbon emissions [7]. Thus, **we need to make datacenters sustainable**. Particularly, datacenters need to reduce their *embodied emissions* (emissions from manufacturing, transporting, and disposing hardware), as these emissions will account for *82% of emissions* [6, 13]. My work **identifies storage as a major and overlooked factor in embodied emissions** [10, 11]. I have developed systems to reduce these emissions, **including >50% emission reduction for flash caching.** For my work, I have received **2 best paper awards**. There is plenty more work needed to ensure sustainable datacenters, and my future work plans to continue addressing this pressing problem.

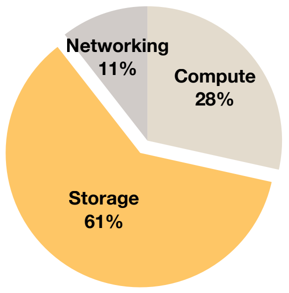

Fig. 1: Breakdown of embodied emissions at Azure [10, 13].

As datacenters increase their renewable energy usage, *operational emissions*[1] (emissions from producing energy used in datacenters) will decrease. Thus, embodied emissions will dominate datacenter emissions. Within embodied emissions, I found that storage overshadows other sources (Fig. 1). **Datacenters need to focus on reducing storage emissions.**

My work enables sustainable datacenters by reducing storage emissions. By rethinking assumptions about storage interfaces, understanding underlying hardware trends, and grounding system design in mathematical modeling, my research enables denser storage devices to stay in datacenters for longer — reducing emissions while maintaining performance. So far, my work has addressed:

1. **Efficient caching of tiny objects in flash** (SOSP 2021 Best Paper) [8, 9]: Tiny objects (≈100 bytes) are prevalent in applications like social graphs, but their size causes either too many writes or too much memory for flash caches. Kangaroo is a flash cache tailored for billions of tiny objects that **reduces misses by 29% and writes by 38% in production**.
2. **New flash interfaces to reduce emissions** (OSDI 2024) [11]: While a large improvement, Kangaroo has too many writes for long-lived, dense flash. These uncontrollable writes are from *within* the device. I introduce a class of interfaces, *Write-Read-Erase iNterfaces (WREN)*, that expose fundamental flash constraints and allow for control over *all* writes. *FairyWREN* leverages WREN to **reduce writes by 12.5x and flash emissions by 33%** over Kangaroo.
3. **New Declarative IO interface for distributed storage**: Distributed storage runs on hard drives. New drives have increasingly higher capacities, without an equal IO increase. Distributed storage needs to reduce their IO to use these devices, particularly from maintenance tasks such as scrubbing and transcoding. These tasks cause the majority of disk IO. I introduce Declarative IO, an interface that exposes inherent flexibility in these tasks to create IO overlap. This new interface **reduces maintenance reads by 60%**.

My work has appeared in top venues such as OSDI [2, 11] and SOSP [8] and **received two best paper awards (SOSP 2021 [8], SIGCSE 2023 [5])**[2]. In addition, I received a 2021 NDESG fellowship, was named a 2023 EECS Rising Star, and am a 2025 Siebel Scholar. I have collaborated frequently with industry including with Meta, Microsoft, Google, and Western Digital. I plan to continue these collaborations.

Looking forward, I plan on pushing system design to further reduce datacenter emissions. I will examine how storage assumptions, such as fixed device capacity, unnecessarily limit lifetime and density. I will investigate enabling low-emissions, high-throughput storage solutions as machine learning workloads stress the current storage architecture. And finally, I will explore how to better quantify carbon emission while also expanding understanding of what makes a sustainable datacenter beyond only carbon-focused metrics.

[1] Operational emissions are Scope 2 and embodied emissions are Scope 3, according to the greenhouse gas protocol.

[2] See my teaching statement for details on SIGCSE paper [5].

1 Reducing Emissions in Flash Caching with Kangaroo and FairyWREN

To balance performance and cost, data centers employ *caches*, storing popular data in fast, expensive memory and the rest in slow, but cheap disks. At scale, major internet companies use flash for caches, which is cheaper than DRAM and faster than hard disk. Flash caches need to overcome two main challenges to be sustainable: (1) they need to not introduce large DRAM indexes, since DRAM has both high operational and embodied emissions, and (2) they need to have no extra writes, since flash wears out with writes. Unfortunately, prior flash caches either used too much DRAM or had too many writes. My project, Kangaroo [8, 9], is the first step toward a sustainable flash cache: enabling low-DRAM, low-write flash caches. The remaining extra writes in Kangaroo are hidden behind the traditional flash interface. FairyWREN [11] explores flash interfaces to reduce these device writes. Together, these two systems cause a **>50% reduction in cache emissions** over the prior state-of-the-art without hurting cache miss ratio or throughput.

Kangaroo: Efficiently caching tiny objects in flash (SOSP 2021 Best Paper). Social networks, microblogs, and emerging sensing applications in the Internet of Things (IoT) need to access tiny objects such as graph edges, text, and metadata. However, tiny objects are uniquely challenging for flash caches, because the objects are orders-of-magnitude smaller than the write size of flash, causing either write endurance problems or requiring unreasonable memory overheads.

Prior flash caches. Existing cache designs either require too many writes or too much memory. Log-structured caches [4] minimize writes by storing all objects sequentially, achieving close to the minimum writes. Unfortunately, these caches need a full index to quickly find objects, which, even when highly optimized, requires too much DRAM to be sustainable. Set-associative caches [2] minimize DRAM indexing by mapping objects to fixed size locations, *"sets"*, on flash using a hash of the object's key. However, set-associative caches suffer from too many writes: they have to write 4 KB for every 100 byte object, causing a *write amplification* of 40x. These additional writes lead to early wear-out due to flash's limited write endurance.

Kangaroo. To optimize both writes and DRAM usage, I developed a hybrid flash cache called Kangaroo that combines prior designs to get the best of both worlds. Kangaroo's main insight is that a small log-structured cache (KLog) minimizes write amplification by amortizing set writes in a large set-associative cache (KSets) over multiple objects. Kangaroo first writes to KLog, taking advantage of its low write amplification. Once KLog fills, Kangaroo flushes each object in a log segment to the set in KSets corresponding to the key's hashed value along with *all other objects in KLog that map to the same set.* Essentially, Kangaroo reduces writes by creating hash collisions. Since Kangaroo is a cache, if there are not enough collisions for a set, objects are evicted. Kangaroo's design has a very low memory overhead of 7.0 bits/obj, a 4.3x improvement over prior work.

Thus, Kangaroo has both low memory overhead and low write amplification, unlike prior systems. Avoiding both of these pitfalls causes Kangaroo to have 29% fewer misses under realistic workloads. **My deployment of Kangaroo on production servers at Meta showed a 38% reduction in writes compared to their production tiny-object cache.**

FairyWREN: New flash interfaces for sustainable caching (OSDI 2024). While Kangaroo greatly reduces writes without a large memory overhead, long-lived and dense flash requires further write reductions. Unfortunately, Kangaroo cannot address the main remaining source of writes: rewrites inside the flash device. Theses writes exist because flash's current interface, *Logical-Block-Addressable Devices (LBAD)*, allows 4 KB writes even though flash's erase granularity is closer to a gigabyte (flash must erase before overwriting data). This mismatch requires the device to reclaim space through garbage collection. Garbage collection leads to uncontrollable writes, particularly bad in caches where every write is a decision on whether to keep objects or evict them.

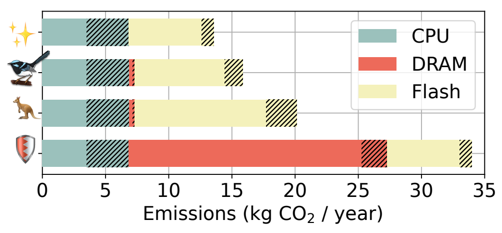

Fig. 2: Yearly carbon emissions for 4 caching systems: optimal emissions with no WA or DRAM (✨), FairyWREN (🐦), Kangaroo (🦘), and an optimal log-structured cache (🛡️). The results include the embodied and operational (hatched) emissions from CPU, DRAM, and flash.

Flash interfaces. New emerging flash SSD interfaces, such as ZNS [3] and FDP [1], allow closer integration of host-level software and flash management. Importantly, these interfaces expose erase operations. I introduce Write-Read-Erase iNterfaces (WREN), an interface categorization that captures the necessary operations for sustainable flash caches. However, WREN does not immediately reduce writes — it only gives the cache control of all writes.

FairyWREN. FairyWREN combines previously device-controlled garbage collection and the caching logic in Kangaroo into one operation called *nesting*, reducing writes. With several other optimizations, FairyWREN decreases writes 12.5x over Kangaroo. This write reduction translates to a 33% decrease in flash emissions over Kangaroo and a >50% reduction in total emissions over log-structured caches, the prior state-of-the-art (Fig. 2).

2 Reducing Emissions in Bulk Storage through Declarative IO

Datacenters need to reduce embodied emissions, the majority of which comes from storage. Most storage devices are in bulk storage, where datacenters use hundreds of thousands of hard disk drives. These hard disk drives' capacities are greatly increasing due to new technology — from <20 TB in 2020 to an expected >50 TB in 2025. Deploying denser drives promises to reduce embodied emissions. Unfortunately, these drives' bandwidth and their IOPS per drive has not increased to match their capacity. This *"IO wall"* prevents the adoption of denser drives. To deploy these more sustainable, denser drives, bulk storage needs to reduce IO.

Most bulk storage IO is maintenance. Surprisingly, most IO does not come from storage users, since most of this user IO is highly cacheable. Rather, *my work identifies maintenance tasks as the primary source of IO in bulk storage*, based on discussions with hyperscalars including Google, Meta, and Microsoft. These tasks are essential, because they ensure data is stored accessibly and reliably with little space overhead. For instance, scrubbing requires reading every byte of data every few months to ensure the data still exists. Decreasing the frequency of scrubbing increases the risk of data loss, an unacceptable outcome for a storage system. Making the problem worse, maintenance tasks come from throughout the datacenter: from databases (e.g., compaction, table statistics), to object stores (e.g., transcoding, object checksums), to distributed file systems (e.g., scrubbing, rebalancing, reconstruction), to disks (e.g., garbage collection).

Insight: Imperative IO destroys flexibility. Today, maintenance tasks use an imperative interface that allows tasks to request only a *specific* piece of data *now*, i.e., read this block now. This unnecessarily limits these tasks' inherent time and order flexibility. For instance, scrubbing must currently be implemented via sequential requests to read individual blocks rather than a directive to read a set of blocks at some time in the next month. If the imperative interface did not force these tasks to eliminate their flexibility, there is massive potential for overlap *between* these tasks. By coordinating IO, storage systems could issue a single command to accomplish multiple tasks.

Declarative IO: Exposing and exploiting IO overlap to enable denser HDDs. To reduce bulk storage IO, I introduce a new interface, Declarative IO, that allows maintenance tasks to declare that they need to read a set of data by a target deadline. Returning to the scrubbing example, rather than writing a large loop through all data, declarative IO allows scrubbing to declare that it needs all data every couple months. Other tasks, such as re-encoding and garbage collection, can do the same. Then, when Declarative IO's *IO Planner* decides it is a good time to read a piece of data for garbage collection, scrubbing and any other declarative tasks get that data *for free.* This new interface *reduces read IO by 60%* relative to imperative IO while running 7 maintenance tasks over a week because each read now performs work for multiple maintenance tasks. Even better, the benefit increases as more maintenance tasks use declarative IO.

3 Future Work

There is, of course, plenty more research to do to ensure sustainable datacenters, both in storage and in datacenter systems more broadly. In general, reducing carbon requires reducing embodied emissions, reducing operational emissions, and better understanding of sustainability in datacenters. My future work will address all three of these aspects.

Enabling denser drives in bulk storage. Storage dominates embodied emissions and, thus, will continue to be a large focus of my continued research agenda. Today, reads are an essential bottleneck to overcome for HDDs. However, as flash becomes cheaper, datacenters will deploy more flash SSDs into bulk storage. Thus, datacenters will need to continue to reduce both read and write IO to bulk storage to permit deployment of dense drives. I identify two new directions to enable these devices: adapting IO from users to reduce read IO and overlapping writes to reduce write IO.

Seamlessly reducing user IO. User tasks such as ML training or analytics are a large source of IO. These applications have two main challenges: they have demanding performance requirements and their developers have little knowledge of the storage stack. Machine learning training, for example, keeps consuming larger training sets, particularly for video generation, stressing storage infrastructure. Promisingly, there is a large amount of order flexibility from changing the ordering of batches *without changing batch content*, so these applications could benefit from a declarative interface. However, Declarative IO cannot become the bottleneck for training, extend the time to converge, nor require extensive storage knowledge to integrate. I will investigate how to reduce ML reads, an area I am confident I can tackle based on my prior work in ML systems [12].

Enabling flash for bulk storage. Writes fundamentally limit flash because flash no longer retains data after too many writes. Each flash generation makes this problem worse. Thus, datacenters need to reduce writes in bulk storage to deploy dense flash. Unfortunately, writes are harder to reduce than reads because they change data. Similar to FairyWREN combining writes between layers, distributed storage can combine different types of writes, such as adding new data, moving data, and regrouping data. Importantly, regrouping and moving data writes do not conflict. I will investigate the interface for combining writes, which will require carefully specifying each write's goal. The resulting system could also incorporate lifetime so new data can also be more intelligently placed, reducing regrouping writes.

Enabling longer lifetimes through rethinking failure. Storage devices do not usually fail completely, rather software decides that they fail when too much of a device's capacity becomes unusable. As storage devices become larger (moving from 10s of TBs to PBs), losing a device because some of its capacity no longer works causes a larger impact. However, devices today have no partial failure state because software assumes that devices' capacities are fixed. This assumption artificially decreases the lifetime and prevents adaptive flash bit-density.

Extending lifetime through partial failures. Both HDDs and SSDs often do not fail fully. Rather, many devices fail due to partial failures, from write heads no longer working (a problem made worse with new HAMR technology) to bad blocks forming on flash. Drives overprovision capacity to handle these failures. Once more capacity is lost than this pre-determined overprovisioning, the device fails. However, as devices get larger, this strategy leaves increasingly more wasted capacity that could continue to be used if devices did not have to adhere to a fixed capacity. My future work will analyze these types of partial failures to determine their frequency and remaining capacity. I will investigate how to expose these capacity changes to distributed storage and how distributed storage can manage partial failures, while mitigating any IO or power overheads.

Adaptive bit-density enables denser, long-lifetime flash. Traditionally, flash SSDs come with a specified write endurance based on the flash cells having a fixed number of bits per cell. Denser flash's lower emissions-per-bit has a trade-off: lower write endurance. Density versus endurance is a false dichotomy. Based on discussions with several manufacturers, flash cells can operate in lower-density modes after they "wearout" at higher densities. Thus, devices could dynamically transition to lower densities as they wear out, removing the conflict between density and lifetime.

Enabling adaptive bit-density requires being able to change device capacity. In addition to supporting variable capacity, I will investigate the emissions impact of, and system support for, adaptively configuring bit-density. While flash cells support different bit-densities, I need to quantify write endurance for pseudo, low-density cells after they were used as higher density cells. I will model this relationship based on experimentation (and in collaboration with SSD manufacturers). I will also explore systems solutions that allow for gradual capacity reduction and other approaches for minimizing the writes needed to transition between bit-densities. I have already started exploring this direction with Sophia Cao, a CMU undergraduate student whom I mentor. Also, adaptive bit-density results in better performance *as the device wears out*, enabling new applications.

Building system-level choices on datacenter sustainability. To truly build sustainable datacenters, we need to better understand where carbon emissions come from and how to reduce them so that systems designers can include sustainability as a first-order metric when considering future computer systems. We also need to expand on the metrics we use to measure sustainability.

Better metrics and transparency. Even within carbon emissions, a relatively well-studied area, finding up-to-date estimates with reasonable uncertainty that allow comparison between different types of the hardware and operational/embodied emissions is non-trivial. For instance, my 2024 HotCarbon paper on storage emissions in the datacenter [10] was the first to identify the full impact of storage on datacenter emissions — rather than just focusing on compute. This knowledge is important so that the research community can focus on projects that have the most impact.

However, there is still too little understanding of carbon emissions as a metric. I plan on developing a better understanding of where cost and emissions are synonymous and where they are not. For example, GPUs are costly but having relatively little embodied emissions, particularly when compared to storage, but finding accurate numbers for this comparison is a challenge. My connections across the industry, from device manufacturers to hyperscalars, will enable me to build better carbon models. I also plan on increasing the transparency of emissions metrics and the breakdown of what contributes to varying emissions. Right now, for instance, it is hard to understand why and how embodied emissions changes between different generations of flash manufacturing.

Moving beyond carbon emissions. Finally, I believe the research community needs to move past just considering carbon emissions as the only sustainability metric. Computing inherently involves chemicals, water, and rare minerals whose use affects our world. Adding more metrics such as water use will lead to a larger trade-off space for datacenter systems that needs to be understood. For instance, building a datacenter in a desert means paying for the environmental impact of developing pipelines for water, but fortunately has an abundance of solar energy. I believe the systems community is the place for this to happen given that systems already have to consider a large trade-off space between different performance metrics, cost, and other overheads. I plan on developing this understanding and how systems-level design impacts a more holistic view of sustainable datacenters and computing more broadly.

References

[1] Overcoming the write amplification problem with nvm express® flexible data placement. https://nvmexpress.org/nvmeflexible-data-placement-fdp-blog/. Accessed: 2023-07-21.

[2] Benjamin Berg, Daniel S. Berger, Sara McAllister, Isaac Grosof, Sathya Gunasekar, Jimmy Lu, Michael Uhlar, Jim Carrig, Nathan Beckmann, Mor Harchol-Balter, and Gregory G. Ganger. The CacheLib caching engine: Design and experiences at scale. In *USENIX OSDI*, 2020.

[3] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, and George Amvrosiadis. ZNS: Avoiding the block interface tax for flash-based SSDs. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 689–703. USENIX Association, July 2021.

[4] Assaf Eisenman, Asaf Cidon, Evgenya Pergament, Or Haimovich, Ryan Stutsman, Mohammad Alizadeh, and Sachin Katti. Flashield: a hybrid key-value cache that controls flash write amplification. In *USENIX NSDI*, 2019.

[5] Bailey Flanigan, Ananya A. Joshi, Sara McAllister, and Catalina Vajiac. Cs-jedi: Required dei education, by cs phd students, for cs phd students. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, 2023.

[6] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 854–867. IEEE, 2021.

[7] Nicola Jones et al. How to stop data centres from gobbling up the world's electricity. *Nature*, 561(7722):163–166, 2018.

[8] Sara McAllister, Benjamin Berg, Julian Tutuncu-Macias, Juncheng Yang, Sathya Gunasekar, Jimmy Lu, Daniel S. Berger, Nathan Beckmann, and Gregory R. Ganger. Kangaroo: Caching billions of tiny objects on flash. In *ACM SOSP*, 2021.

[9] Sara McAllister, Benjamin Berg, Julian Tutuncu-Macias, Juncheng Yang, Sathya Gunasekar, Jimmy Lu, Daniel S. Berger, Nathan Beckmann, and Gregory R. Ganger. Kangaroo: Theory and practice of caching billions of tiny objects on flash. 2022.

[10] Sara McAllister, Fiodar Kazhamiaka, Daniel S Berger, Rodrigo Fonseca, Kali Frost, Aaron Ogus, Maneesh Sah, Ricardo Bianchini, George Amvrosiadis, Nathan Beckmann, et al. A call for research on storage emissions. In *HotCarbon Workshop on Sustainable Computer Systems 2024*, 2024.

[11] Sara McAllister, Yucong "Sherry" Wang, Benjamin Berg, Daniel S. Berger, George Amvrosiadis, Nathan Beckmann, and Gregory R. Ganger. FairyWREN: A sustainable cache for emerging Write-Read-Erase flash interfaces. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.

[12] Foteini Strati, Sara Mcallister, Amar Phanishayee, Jakub Tarnawski, and Ana Klimovic. Déjàvu: KV-cache streaming for fast, fault-tolerant generative LLM serving. In *Proceedings of the 41st International Conference on Machine Learning*, ICML 2024, 2024.

[13] Jaylen Wang, Daniel S. Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, Jonathan Stern, Ricardo Bianchini, and Akshitha Sriraman. Designing cloud servers for lower carbon. In *ISCA*, 2024.