The Storage Wall in LEO Compute Clouds

Vaibhav Bhosale
Georgia Institute of Technology
Atlanta, USA
vbhosale6@gatech.edu

Ada Gavrilovska
Georgia Institute of Technology
Atlanta, USA
ada@cc.gatech.edu

Ketan Bhardwaj
Georgia Institute of Technology
Atlanta, USA
ketanbj@cc.gatech.edu

Abstract

Low Earth orbit (LEO) satellites increasingly carry commodity compute and storage, forming a LEO compute cloud. A satellite is overhead for only a few minutes, so an application that must keep serving a region hands off to a new satellite hundreds of times a day. For a stateful application this handover is a storage event, because the application state has to be written to the next satellite's drive before it can resume there. We argue that flash endurance is a binding constraint on stateful LEO compute. Our model projects that full-state checkpointing at orbital handover cadence exhausts a commodity drive's write endurance in months to a few years. That is shorter than the five-year deployment lifetime of modern constellations, which cannot be serviced once in orbit. A more durable drive raises the endurance budget in proportion to its rating, but at a significantly higher cost. The result is a three-way tension we call LDW, for latency, state durability, and SSD wear. We observe that a stateful design can hold only two of the three, and present a research agenda for designs that exercise the trade-offs across all three axes.

CCS Concepts

• **Information systems** → **Storage management**; *Storage architectures.*

Keywords

LEO Compute Cloud, Orbital Edge Computing, Flash Endurance

ACM Reference Format:
Vaibhav Bhosale, Ada Gavrilovska, and Ketan Bhardwaj. 2026. The Storage Wall in LEO Compute Clouds. In *18th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '26), September 28–29, 2026, Prague, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3837053.3837355

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
HotStorage '26, Prague, Czech Republic
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2902-7/26/09
https://doi.org/10.1145/3837053.3837355

1 Introduction

Over the past decade, Low Earth orbit (LEO) satellites have come to carry commercial off-the-shelf compute and storage, turning these mega-constellations into a LEO compute cloud [2, 6, 10, 15, 16]. This cloud now runs applications from Earth-observation pipelines and content delivery to large language model (LLM) inference [12, 18, 26, 34, 40]. Commercial efforts are propelling this shift, placing machine-learning accelerators in orbit and proposing satellite constellations as data centers [2, 15]. Many of these applications are stateful, holding mutable application state on the satellite. The north star for onboard compute is to keep an application one hop from its users to avoid transmitting data over a scarce downlink. A LEO satellite, however, remains accessible from a fixed region for at most 4.5 minutes [11]. Therefore, an application that must keep serving that region cannot stay on one satellite. To ensure application availability, it must therefore migrate to successive satellites through an application handover, which is the predominant way to mask the motion of the infrastructure [7, 10].

In this paper, we argue that an application handover in a LEO cloud is not only a compute or network event, as in a terrestrial cloud, but also a storage event. The application state a satellite holds for its users must reach the next satellite before it can serve them, because a fresh satellite holds no prior copy of it. Some state can be refetched or recomputed at the destination rather than shipped, but for mutable state that may not always be an option: an in-orbit aggregator would have to replay sensor inputs it no longer holds, and an interactive LLM session would have to re-run prefill over the entire conversation, impacting critical performance metrics like time-to-first-token [25, 41]. Each handover therefore checkpoints the application state and writes it to the next satellite's drive, once every few minutes. Prior work treats this handover as an orchestration problem, met with checkpoint-based migration [10] or replicated state [28], and does not account for the resulting storage (flash) writes, which are bounded by endurance because each cell tolerates only a limited number of writes over the lifetime of the drive. A full-state write at every handover exhausts a commodity drive's endurance well inside the deployment lifetime of a constellation, which cannot be serviced once in orbit.

Concretely, a single commodity 1 TB drive exhausts its rated endurance in under a year, because the high handover

frequency forces it to absorb heavy writes. An application in a deployed LEO compute cloud is handed over 432 times a day [10], and existing checkpoint mechanisms force the drive to absorb more than 3 TB of writes each day. This is several full-drive writes a day, far above the rate a capacity-oriented quad-level cell (QLC) part is rated to sustain [36]. *We project the drive's endurance to run out in about ten months*, far short of the multi-year operating life of a constellation whose drives cannot be swapped once in orbit. The write stream is not confined to one satellite, since the workload runs on every satellite serving it, so the projected wear extends across the entire constellation, not just a few satellites.

A more durable drive does not solve this cheaply. A higher cycle rating raises the endurance budget in proportion, but the write rate is set by the orbit and the workload, not by the drive. A part durable enough to last through the five-year constellation lifetime requires expensive higher-grade flash [1]. The endurance shortfall is therefore a storage-systems design problem, not a hardware one.

We observe that the heavy writes driven by handovers create a three-way tradeoff. The single write decision at each handover sets these three quantities at once, and no choice can satisfy them together. Two of them are what an application provider needs to serve its users: low latency, which keeps application state on the serving satellite, and a durable session, which requires that application state survive the move and any fault. The third is the cost the operator bears for either of them, namely the wear on the serving satellite's solid-state drive (SSD), which is the same endurance the write stream above exhausts. Wear is therefore not a separate cost but the price of keeping application state both local and durable. We name this tension LDW, for latency, state durability, and SSD wear. The severity of the conflict scales with the application state each handover carries, which can vary by orders of magnitude, so the right response depends on the workload deployed.

This paper makes three contributions. (i) A first-principles model of SSD wear under orbital handover (§3) that turns drive lifetime into a function of the orbit and the workload. (ii) A framing of the resulting tension, LDW, with prior approaches placed at its corners (§4). (iii) A research agenda for designs that trade partial latency, durability, and wear across the three axes (§5). We argue that flash endurance bounds stateful LEO compute, and that designing within that bound is a storage-systems problem our community must solve for the LEO compute cloud to be viable.

2 Background and Related Work

LEO Compute Clouds. A LEO satellite orbits Earth at altitudes between 200 and 1600 km, moving at roughly 27,000 km/h in orbit [4]. These satellites now carry commodity

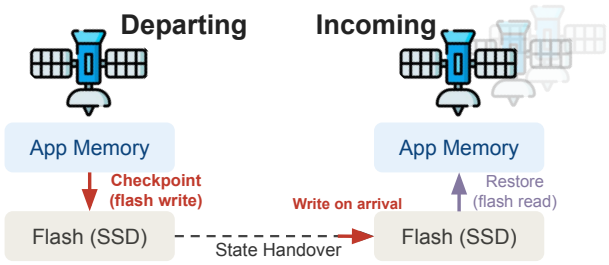

Figure 1: A stateful handover is a storage operation. As a satellite moves out of range of a region, it checkpoints application state from memory to flash and sends it to the next satellite, which writes it on arrival and restores it before serving the same region. Both writes repeat for every handover.

compute and storage, from commercial off-the-shelf processors flown on satellites [18, 39] to a data-center-class GPU recently operated in orbit [15]. A constellation consists of thousands of LEO satellites providing continuous coverage of the Earth, organized into shells at different altitudes and inclinations. Inter-satellite links (ISLs) connect satellites within a shell, so traffic between shells routes through a ground station. This onboard capacity, organized into a managed, cloud-like service, is the LEO compute cloud, where applications run on the satellites and serve the users on Earth [6, 10].

The key difference from a terrestrial cloud is that the infrastructure is constantly moving. A satellite is visible for at most 12 minutes from a point on Earth [11]. However, to maintain the elevation angle required for a usable link, a satellite remains accessible for a maximum of 4.5 minutes [11]. An application that must keep serving a fixed region therefore cannot stay on one satellite, and it hands over to the next satellite covering that region as the current one departs. These handovers maintain application availability. Without them, every request would go through additional ISL hops to reach the original satellite, raising latency even above that of serving the request from the ground.

What a handover costs depends on how the system performs it. In our prior work Krios, we observe that an application stays on a satellite for a median 200 seconds, giving 432 handovers a day [10]. Prior work addresses this cost with orchestration systems that mask the impact of motion by handing applications from one satellite to the next, maintaining the *geostationarity* of applications [7, 10, 28], so that an application appears stationary to users on the ground even as the satellite hosting it changes. The cost that remains is measured in network bytes and latency. Krios hides the application initialization time by starting the handover before the serving satellite moves out of range [10], so the longer a handover takes, the larger the resource footprint for every application deployed in a LEO cloud.

For a stateful application, however, the handover is also a storage operation. Moving application state directly from

Workload class	State per handover
Sensor aggregation [27, 28]	~2 kB [23]
Web / CDN cache [12, 40]	~1–100 GB [8, 40]
LLM serving, KV-cache [34]	~2.6 GB

Table 1: Application state per handover for different LEO cloud application classes.

memory to memory would avoid the write, but both instances have to run for the duration of the transfer, and that duration grows with the state size [28]. Checkpoint/restore tooling [38] instead separates when the state is sent from when the application resumes on the destination [10]. That checkpoint is a write, so every handover incurs at least one checkpoint (Figure 1).

Prior approaches do not account for the impact of this write. In most current designs, a handover writes the full application state. We categorize them into three options. Read-mostly applications can refetch content at the destination, trading the write for latency on every miss [40]. The second keeps the application and its state resident on every satellite they may run on, at least 475 per application [10], which trades the write for the cost of keeping those copies current: each update writes 475 copies against the handover path's 432 per day, so it pays only for state that changes less than about once a day, and holding that many copies bounds how many applications the constellation can host. The third ships only the change, as terrestrial migration does against a base snapshot the destination already holds, but a satellite the application ran on earlier holds at best a stale copy, and an application returns to a given satellite only after cycling through many others. None of these routes applies to applications with mutable state that must be migrated at every handover. For these workloads the write is unavoidable, and flash wear becomes the cost that limits the drive's lifetime.

Workloads and Storage Cost. The LEO compute cloud runs a range of workloads (Table 1): content delivery and web serving [12, 26, 40], IoT and sensor aggregation [8, 28], machine-learning inference on captured imagery [17, 18], and LLM serving [34]. A workload's handover cost depends on its application state, the in-memory working set, session context, and on-disk state that a running instance must carry with it. The workloads that incur the full write hold application state that is large, mutable, and unique to a long session, such as key-value and session stores, or the growing key-value cache of an LLM serving a conversation [34]. These state sizes span kilobytes to hundreds of gigabytes. Sensor aggregation holds kilobytes per pass, while a single 8k-token session on a 70B-parameter model with grouped-query attention works out to roughly 2.6 GB of key-value cache at half precision, and a few hundred cached web pages at a 2.86 MB median run to gigabytes [20]. Importantly, not all

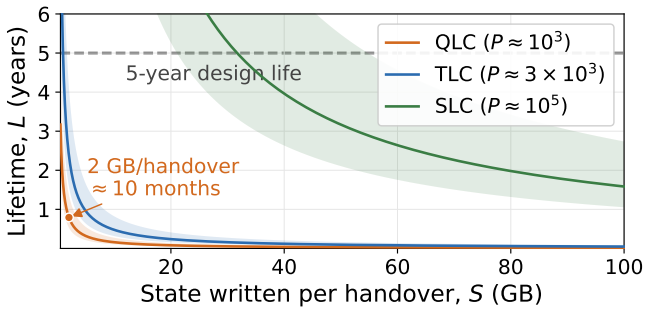

Figure 2: Projected drive lifetime (Equation 1) versus application state written per handover, for three commodity endurance classes. The dashed line marks the five-year design life of the constellation. Shaded bands span handover rates of 250 to 650 per day with the plot line using 432.

large state is costly. An observation satellite captures more data than it can downlink [17, 18], but writes it once in long sequential streams, which consumes little endurance. Neither prior compute work nor Earth-observation workloads account for the cost we identify: mutable application state written to flash at every handover, hundreds of times a day.

Flash Endurance in the Space Environment. A NAND flash cell tolerates only a bounded number of program/erase cycles before it can no longer hold charge reliably. This endurance budget shrinks as cells store more bits. It is on the order of 50,000 to 100,000 cycles for single-level cell (SLC) flash and a few thousand for the triple-level cell (TLC) parts in commodity drives [13], falling to around 1000 for the dense QLC parts a capacity-oriented deployment favors [32]. Rewriting a flash cell requires first erasing it, and each program/erase cycle spends part of that endurance budget, so a drive wears out as writes accumulate. Radiation also degrades flash, though most LEO satellites orbit below the Van Allen belts and within the Earth's magnetic field, which limits the radiation dose the satellite experiences. LEO missions accordingly fly radiation-tolerant, industrial-grade, or screened commercial flash in place of radiation-hardened space-grade parts [5]. Radiation-tolerant parts cost several times as much as commercial parts, and radiation-hardened parts can cost a hundred times as much [1]. Radiation hardening also enlarges the circuitry, lowering storage density [19]. Modern constellations deploy satellites on a five-year horizon [3]. They rely on commodity hardware to lower costs, so they often opt for the cheaper, dense, low-endurance QLC drives. For a commodity constellation, the endurance budget is set by the commodity flash it can afford, and raising it requires additional capacity, power, or cost.

3 SSD Wear Model in Space

A flash drive's endurance is a fixed budget of writes, and each handover spends it at a rate set by the orbit and the workload. We account for writes in host bytes throughout, the bytes an

application issues to the drive. A drive of capacity C whose cells are rated for P program/erase cycles can absorb $E = C \cdot P$ host bytes before it wears out [13]. Vendors publish this budget as a terabytes-written (TBW) or drive-writes-per-day (DWPD) rating, both measured in host bytes [13, 21].

Each handover persists application state of size S. As the new satellite may not hold any earlier copy of that state, a handover requires a full checkpoint rather than the incremental checkpointing terrestrial systems rely on [30]. A write-inflation factor α captures the host bytes a handover adds on top of S, and two effects set it. The checkpoint is written to disk at both the departing and the arriving satellite. With data journaling enabled, the filesystem writes each byte twice, once to the journal and once to its final location [31]. Together these give $\alpha = 4$. We exclude garbage collection, because the TBW and DWPD ratings we compare against are quoted in host bytes and already include the resulting write amplification [21]. Handovers arrive at rate H per day. A satellite hosting N applications, each handing off at that rate, therefore absorbs $W = H \cdot S \cdot \alpha$ host bytes per day, a per-drive rate, where $S = N \cdot s$ aggregates the state of all N residents, each of size s. Lifetime is then the endurance budget divided by the rate:

$$L = \frac{E}{W} = \frac{C \cdot P}{H \cdot S \cdot \alpha}. \quad (1)$$

In Equation 1, lifetime is inversely proportional to the handover rate, the application state per handover, and the inflation. We refer to the consumption of E at this rate as wear. None of the three terms in the denominator is a property of the drive. H is set by the orbit, and S and α are set by the workload and the handover mechanism. A more durable drive can raise P and scale L up in proportion, but the inverse dependence on H, S, and α remains unchanged.

To put this in perspective, consider a commodity 1 TB drive built from QLC flash ($P \approx 10^3$ [32]) on a satellite that writes $S = 2$ GB of application state at each handover[1], with the inflation set to $\alpha = 4$. At the $H \approx 432$ handovers per day reported by our prior work Krios [10], the drive receives $W \approx 3.4$ TB of host writes per day. That is more than three drive-writes per day, against the roughly 0.6 drive-writes per day a capacity-oriented QLC part is rated to sustain across its five-year warranty [36]. This accounts for handover writes alone. The writes an application issues while serving requests are additional, so W is a lower bound. At this rate, *we project the $E = 1$ PB budget to run out in about ten months, and a more durable TLC part ($P \approx 3 \times 10^3$) to last only a little over two years.* Both fall well short of the five-year deployment lifetime of the constellation.

[1] $S = 2$ GB is a deliberately conservative operating point, below even a single LLM session (Table 1); we sweep upward in Figure 2.

Wear Model Sensitivity. Of the five inputs to Equation 1, C and P are fixed for a given drive and H by orbital motion. We model one representative drive per endurance class, since drives within a class cluster within roughly a factor of two to three. The other two inputs belong to the handover. We sweep S from 0.5 to 100 GB at $\alpha = 4$ in Figure 2, because S spans the widest range across workloads. QLC remains below the five-year constellation lifetime across the sweep. TLC exceeds it only below roughly 1 GB, and SLC below roughly 30 GB, though at a much higher cost. A storage-system design can primarily affect S and α, so the size of the state written per handover and the handover mechanism become important system-level knobs. H itself is an upper bound: a satellite that carries its application load for only a fraction ρ of the day writes at ρH, and lifetime scales as $1/\rho$. Even at $\rho = 1/3$, *we project the QLC drive to last 2.4 years.* C and P can be increased as well: a larger drive, a higher-endurance part, or more drives per satellite all raise E, and a shorter replacement cycle limits the exposure, but each buys endurance with capacity, power, or launch mass the deployment may need elsewhere.

4 The Trilemma and the Design Space

Consider again a model-serving session, whose key-value cache grows in RAM as the conversation lengthens. A serving engine moves that cache out only under capacity pressure, swapping blocks to a slower tier [22], so those writes are partial and driven by demand. A handover instead requires a checkpoint, so the whole cache moves onto the drive on a schedule set by the orbit rather than by memory pressure. Rebuilding it with prefill at the destination only trades that write for a cost that recurs at every handover. Requiring the application state to survive the handover and stay local for latency therefore forces it onto the drive, and the resulting write spends endurance. That one decision, repeated every few minutes, is where the conflict lies.

The LDW Trilemma. A single handover decision sets three quantities at once: latency to the users, durability of the application state across the move, and SSD wear on the serving satellite's drive. We call this three-way tension LDW. At most two can be met together. Three pure strategies operate at the extremes of this trilemma (Table 2). Local persistence checkpoints the application state to the serving satellite's own drive. The state stays local and survives the move, but the write spends the endurance budget. Remote persistence writes each update synchronously to a fixed home on the ground or a designated satellite. The application state is durable and the local drive is spared, but every write travels a network path, so latency increases. Memory-only keeps the application state in volatile RAM and never persists it. It

Design	Low latency	Durable state	Low wear
Local persistence	✓	✓	✗
Remote persistence	✗	✓	✓
Memory-only	✓	✗	✓

Table 2: The three corners of LDW. Each pure strategy achieves two goals and sacrifices the third (✗). SSD wear is measured on the serving satellite's drive.

is fast and spends no endurance, but a handover or a fault loses the state. Each corner gives up exactly one axis.

Prior LEO Cloud Approaches. Existing designs sit at these corners. State offloaded to the ground is remote persistence, but every access goes through an intermittent satellite-to-ground path [11, 37]. Checkpointing to the serving satellite is local persistence, which spends the endurance budget at every handover. Our prior work Krios [10] sits at that corner, and it optimizes availability and bandwidth across handover rather than the resulting wear. Terrestrial flash management [13] targets a stable workload on a fixed drive rather than full-state rewrites at orbital cadence. None of them treats the volume of state written per handover as something a design can reduce.

5 Research Agenda

No existing LEO cloud system design sits between the three extremes, trading partial latency, partial durability, and partial wear instead of giving up one axis. The terrestrial tools for cheap state movement, namely delta encoding [35] and live migration [14, 29, 33], do not transfer directly, because each rests on assumptions that do not hold in a LEO cloud. Delta encoding needs a base image already waiting at a known destination, but each handover targets a different satellite holding no prior copy. Live migration needs a fast and stable link, but the contact window is short and variable [11], and throughput and latency vary far more than on a terrestrial link [24]. Terrestrial designs rarely budget power against each byte moved, while on a satellite both the radio and the drive draw from a solar supply that varies with the orbit [9]. Most importantly, they treat handover as a rare event worth optimizing occasionally, while in a LEO cloud it occurs every few minutes. Closing this gap calls for several research directions rather than a single system. We sketch four, borrowed from terrestrial systems and rethought for this setting. Each keeps application state off the local drive or shrinks what lands there, and each therefore pushes cost onto the network.

Bounded log shipping. Instead of copying the entire application state, the system can ship a bounded operation log at each handover and replay it at the destination. Only the log is written, which reduces wear. However, this approach is only suitable for the small set of applications whose updates decompose into a compact operation log, such as key-value and session stores. It does not fit large derived state, like the cache an LLM builds over a conversation. Further, unreplayed operations are lost if the source fails, and replay delays when the destination can start serving. The open question is how to compress updates into a log small enough to replay without significantly extending the handover.

Policy-triggered flushing. Rather than persisting at every handover, the system can pass application state directly from memory to memory and write to the drive only when a policy requires it. A write is warranted when the path is lossy, when the application state is large enough that a live transfer would lengthen the handover, incurring a larger resource footprint on the constellation [28], or when the destination lies in a different orbital shell that no ISL path reaches, so the transfer detours through a ground station. Wear then scales with the fraction of handovers that persist. A memory transfer that fails with no persisted copy loses the application state, so the policy trades wear for durability. The choice of destination decides whether a live transfer is possible at all, so this policy cannot sit in the storage layer alone. It is unclear which signals justify skipping the write, and how they should feed into the orchestration logic that selects handover targets.

Hot/cold split. A design can keep the hot working set, the part that latency depends on, local and volatile, and replicate the cold remainder to the ground. Only the cold fraction leaves the satellite and only the hot fraction risks loss, so one workload sits between two corners at once. This suits applications with skewed access, such as a content cache where a small popular set serves most requests, and not those that touch most of their state on every request. Terrestrial tiering reaches its cold tier at a stable cost, while in a LEO cloud a cold access waits for a contact window or routes over ISLs at higher latency, and the hot set has to be rebuilt on each new satellite. A miss can therefore take longer than the application tolerates. How to partition application state online as access patterns shift remains unresolved, since the proximity of ground stations sets the cost of a miss and differs across deployments.

Base-plus-log handover. A base image refreshed slowly, perhaps once per orbit, together with a small log of recent changes, lets the system ship only the log. The destination then holds a base image to apply the log against, so a handover transfers and writes far less than the full application state. The best candidates are applications whose state is mostly stable between handovers, such as a cached dataset or a search index, rather than ones that rewrite most of it each interval. The base image has to be present and kept current on every satellite the application may run on next, at least 475 satellites per application [10]. Whether staging that many copies can cost less in writes than it saves remains an open question.

What to Measure. Evaluating these directions requires a metric per axis. Wear is the rate at which the endurance budget is consumed, in bytes written to the local drive per unit time. The latency cost of avoiding those writes is the added time to serve a request whose application state is either absent from the serving satellite or still being rebuilt. Durability is the window of application state at risk, the updates not yet persisted when a fault occurs. A strategy is then a point (wear, latency, window), and the attainable points form a Pareto frontier that remains to be measured. Two of the three extremes resolve to fixed points. Local persistence writes $H \cdot S \cdot \alpha$ bytes per day and leaves nothing at risk, and memory-only writes nothing and leaves the whole working set at risk. Remote persistence does not resolve to a fixed point, because its latency depends on where the satellites are. Its writes are small, but each transfer crosses a network path whose cost varies with link quality [24] and with the contact window it must fit into [37]. A small update incurs a fixed per-transfer overhead, and a large one is limited by available bandwidth. We propose *bytes written avoided* as the wear coordinate, the drive writes a strategy saves against a local-persistence baseline on the same workload, reported with the latency and window it costs. It can be computed from write counters that systems already keep.

6 Conclusion

An application handover occurs hundreds of times a day in a LEO constellation. Prior work has addressed it as an orchestration concern. In this paper, we have argued that it is also a storage problem, as the resulting write stream can exhaust a commodity drive's endurance in months to a few years. We present a research agenda to identify designs that trade across all three axes of LDW. Fulfilling the promise of a LEO compute cloud will require rethinking storage system designs. This paper is a step in that direction.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback that helped us improve this paper. This project was supported in part by NSF grant CNS-2345827 and a grant from the Space Research Institute at Georgia Tech.

References

[1] 2020. Commercial Space Radiation Tolerance. Military & Aerospace Electronics. https://www.militaryaerospace.com/sensors/article/14178781/commercial-space-radiation-tolerance

[2] Blaise Agüera y Arcas, Travis Beals, Maria Biggs, Jessica V. Bloom, Thomas Fischbacher, Konstantin Gromov, Urs Köster, Rishiraj Pravahan, and James Manyika. 2025. Towards a Future Space-based, Highly Scalable AI Infrastructure System Design. Google Project Suncatcher. arXiv:2511.19468 [cs.DC]

[3] Muaz Ali, Utkarsh Upadhyay, Sean McCormick, Joseph Hill, and Beichuan Zhang. 2026. Starlink Constellation: Deployment, Configuration, and Dynamics. arXiv:2603.25835 https://arxiv.org/abs/2603.25835

[4] Christopher S. Allen, Martina Giraudo, Claudio Moratto, and Nobuyasu Yamaguchi. 2018. Chapter 4 - Spaceflight environment. In *Space Safety and Human Performance*, Tommaso Sgobba, Barbara Kanki, Jean-François Clervoy, and Gro Mjeldheim Sandal (Eds.). 87–138.

[5] ATP Electronics. 2025. Powering Low Earth Orbit (LEO) Satellites with Radiation-Tolerant Storage Solutions. https://www.atpinc.com/blog/low-earth-orbit-satellites-ssd-challenge-solutions

[6] Ketan Bhardwaj, Vaibhav Bhosale, and Ada Gavrilovska. 2025. Ushering in the Low Earth Orbit Compute Cloud Era. *Computer* 58, 10 (2025), 78–86.

[7] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. 2020. In-orbit Computing: An Outlandish Thought Experiment?. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets)*. 197–204.

[8] Vaibhav Bhosale, Ketan Bhardwaj, and Ada Gavrilovska. 2020. Toward Loosely Coupled Orchestration for the LEO Satellite Edge. In *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*.

[9] Vaibhav Bhosale, Ketan Bhardwaj, and Ada Gavrilovska. 2023. Don't Let Your LEO Edge Fade at Night. In *1st Workshop on Hot Topics in System Infrastructure (HotInfra), co-located with ISCA 2023*.

[10] Vaibhav Bhosale, Ada Gavrilovska, and Ketan Bhardwaj. 2024. Krios: Scheduling Abstractions and Mechanisms for Enabling a LEO Compute Cloud. In *Proceedings of the 2024 ACM Symposium on Cloud Computing (SoCC)*. 322–340. doi:10.1145/3698038.3698566

[11] Vaibhav Bhosale, Ahmed Saeed, Ketan Bhardwaj, and Ada Gavrilovska. 2023. A Characterization of Route Variability in LEO Satellite Networks. In *Passive and Active Measurement (PAM)*. Springer, 313–342.

[12] Rohan Bose, Saeed Fadaei, Nitinder Mohan, Mohamed Kassem, Nishanth Sastry, and Jörg Ott. 2024. It's a bird? it's a plane? it's CDN!: investigating content delivery networks in the LEO satellite networks era. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. 1–9.

[13] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. 2017. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. IEEE* 105, 9 (2017), 1666–1704. doi:10.1109/JPROC.2017.2713127

[14] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live Migration of Virtual Machines. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 273–286.

[15] CNBC. 2025. Nvidia-backed Startup Starcloud Trains First AI Model in Space. CNBC News. First data-center-class GPU (NVIDIA H100) operated in low Earth orbit. https://www.cnbc.com/2025/12/10/nvidia-backed-starcloud-trains-first-ai-model-in-space-orbital-data-centers.html

[16] DatacenterDynamics. 2026. SpaceX Files for Million-Satellite Orbital AI Data Center Megaconstellation. DatacenterDynamics. SpaceX proposes scaling Starlink satellites into an orbital data center constellation for AI. https://www.datacenterdynamics.com/en/news/spacex-files-for-million-satellite-orbital-ai-data-center-megaconstellation/

[17] Bradley Denby, Krishna Chintalapudi, Ranveer Chandra, Brandon Lucia, and Shadi Noghabi. 2023. Kodan: Addressing the Computational Bottleneck in Space. In *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Volume 3*. 392–403.

[18] Bradley Denby and Brandon Lucia. 2020. Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System. In *Proceedings of the 25th International Conference on Architectural Support*

for Programming Languages and Operating Systems (ASPLOS). 939–954.

[19] Nikhil Gupta, Bert Vermeire, Hugh Barnaby, Murat Goksel, Edward Li, and David Czajkowski. 2007. Design of a 1 Gb Radiation Hardened NAND Flash Memory. In *Proc. 2007 Non-Volatile Memory Technology Symposium (NVMTS)*. IEEE, 10–14.

[20] HTTP Archive. 2025. Page Weight. The Web Almanac by HTTP Archive. https://almanac.httparchive.org/en/2025/page-weight

[21] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. 2009. Write Amplification Analysis in Flash-Based Solid State Drives. In *Proc. SYSTOR 2009: The Israeli Experimental Systems Conference*. ACM, Article 10. doi:10.1145/1534530.1534544

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. 611–626. doi:10.1145/3600006.3613165

[23] Pooja Lepcha, Tharindu Dayarathna Malmadayalage, Necmi Cihan Örger, Mark Angelo Purio, Fatima Duran, Makiko Kishimoto, Hoda Awny El-Megharbel, and Mengu Cho. 2022. Assessing the capacity and coverage of satellite iot for developing countries using a cubesat. *Applied Sciences* 12, 17 (2022), 8623.

[24] Sami Ma, Yi-Ching Chou, Haoyuan Zhao, Long Chen, Xiaoqiang Ma, and Jiangchuan Liu. 2023. Network Characteristics of LEO Satellite Constellations: A Starlink-Based Measurement from End Users. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*. 1–10. doi:10.1109/INFOCOM53939.2023.10228912

[25] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 118–132. doi:10.1109/ISCA59077.2024.00019

[26] Tobias Pfandzelter and David Bermbach. 2021. Edge (of the Earth) Replication: Optimizing Content Delivery in Large LEO Satellite Communication Networks. In *21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 565–575.

[27] Tobias Pfandzelter and David Bermbach. 2022. Celestial: Virtual software system testbeds for the leo edge. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. 69–81.

[28] Tobias Pfandzelter and David Bermbach. 2024. Komet: A Serverless Platform for Low-Earth Orbit Edge Services. In *Proceedings of the 2024 ACM Symposium on Cloud Computing (SoCC)*. doi:10.1145/3698038.3698517

[29] Maksym Planeta, Jan Bierbaum, Leo Sahaya Daphne Antony, Torsten Hoefler, and Hermann Härtig. 2021. MigrOS: Transparent Live-Migration Support for Containerised RDMA Applications. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 47–63. https://www.usenix.org/conference/atc21/presentation/planeta

[30] James S. Plank, Micah Beck, Gerry Kingsley, and Kai Li. 1995. Libckpt: Transparent Checkpointing under Unix. In *Proc. USENIX Winter 1995 Technical Conference*. USENIX Association, 213–223.

[31] Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2005. Analysis and Evolution of Journaling File Systems. In *USENIX ATC, General Track*. 105–120.

[32] Tianyu Ren, Yajuan Du, Jinhua Cui, Yina Lv, Qiao Li, and Chun Jason Xue. 2025. Device-Level Optimization Techniques for Solid-State Drives: A Survey. *arXiv preprint arXiv:2507.10573* (2025). arXiv:2507.10573 [cs.AR]

[33] Adam Ruprecht, Danny Jones, Dmitry Shiraev, Greg Harmon, Maya Spivak, Michael Krebs, Miche Baker-Harvey, and Tyler Sanderson. 2018. VM Live Migration At Scale. *SIGPLAN Not.* 53, 3 (mar 2018), 45–56. doi:10.1145/3296975.3186415

[34] Thomas Sandholm, Sayandev Mukherjee, Lin Cheng, and Bernardo A. Huberman. 2025. SkyMemory: A LEO Edge Cache for Transformer Inference Optimization and Scale Out. arXiv:2505.14427 [cs.DC]

[35] Constantine P. Sapuntzakis, Ramesh Chandra, Ben Pfaff, Jim Chow, Monica S. Lam, and Mendel Rosenblum. 2002. Optimizing the Migration of Virtual Computers. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

[36] Solidigm. 2024. Solidigm D5-P5336 Product Brief. Product brief. Capacity-oriented QLC data center SSD, rated ≈0.6 DWPD over a five-year warranty. https://www.solidigm.com/products/data-center/d5/p5336.html

[37] Deepak Vasisht, Jayanth Shenoy, and Ranveer Chandra. 2021. L2D2: Low Latency Distributed Downlink for Low Earth Orbit Satellites. In *Proc. ACM SIGCOMM 2021 Conference*. ACM. doi:10.1145/3452296.3472932

[38] Ranjan Sarpangala Venkatesh, Till Smejkal, Dejan S. Milojicic, and Ada Gavrilovska. 2019. Fast In-Memory CRIU for Docker Containers. In *Proceedings of the International Symposium on Memory Systems (MEMSYS)*.

[39] Ruolin Xing, Mengwei Xu, Ao Zhou, Qing Li, Yiran Zhang, Feng Qian, and Shangguang Wang. 2024. Deciphering the enigma of satellite computing with cots devices: Measurement and analysis. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 420–435.

[40] William X. Zheng, Aryan Taneja, Maleeha Masood, Anirudh Sabnis, Ramesh K. Sitaraman, and Deepak Vasisht. 2025. StarCDN: Moving Content Delivery Networks to Space. In *Proceedings of the ACM SIGCOMM 2025 Conference*. 948–962. doi:10.1145/3718958.3754345

[41] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 193–210.