Magellan: A Carbon-Aware Task Scheduler for Geo-Distributed Systems

Will Kaminski *(student)* • **Hari Sundar** *(advisor)* • Tufts University

Motivation and Research Gap

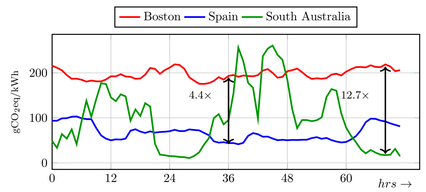
Carbon intensity varies across time and geography. The same computational task can produce substantially different emissions depending on when and where it executes.

Existing approaches leave a systems gap. Prior carbon-aware schedulers commonly:

- delay workloads within a single region,
- evaluate spatial shifting primarily through simulation, or
- support stateless or application-specific workloads.

Research question: Can long-running, stateful workloads be migrated to cleaner regions while accounting for checkpointing, transfer, restore, network energy, performance overhead, and monetary cost?

Global Carbon Opportunity



Location matters. Carbon intensity changes independently across regions, so the cleanest location changes throughout the day.

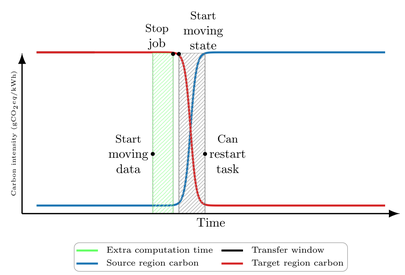
The trace highlights regional carbon-intensity differences of 4.4× and 12.7×. These differences create opportunities to **pause during dirty intervals** or **migrate toward cleaner regions**.

Contributions

- **Dcentralized architecture:** carbon-aware scheduling; each site runs the same peer service.
- **Unified decision model:** jointly models carbon, monetary cost, makespan, and migration overhead with an explicit budget cap.
- **Transparent stateful migration:** supports checkpoint-restart using CRIU for unmodified processes and native checkpoints for LLM tasks and Dendro-GR.
- **Resource-aware arbitration:** uses migration bidding and destination-side admission to share capacity-constrained, heterogeneous sites gracefully.
- **Real-system validation:** evaluates Magellan on CloudLab and Google Cloud under real network and hardware constraints.

Magellan Workflow

Monitor → Score → Prefetch → Checkpoint → Resume



Minimize interruption.

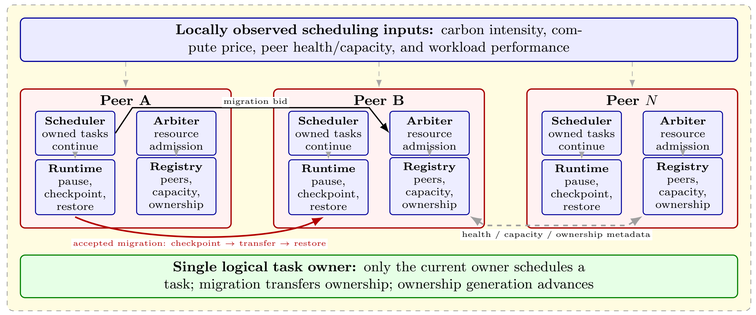
Immutable input data is prefetched while the job continues running.

The task is then paused only long enough to checkpoint and transfer its remaining state.

Execution resumes at the cleaner destination from the preserved checkpoint.

Decentralized System Implementation and Testbed

Fully decentralized peer service. Every site runs the same Python/FastAPI Magellan service. Each peer contains a **Scheduler**, **Arbiter**, **Runtime**, and **Registry**. Source schedulers choose among Continue, Pause, and Migrate; destination arbiters independently admit migration bids subject to measured resource capacity.



CloudLab

Clemson ⟷ Utah ⟷ Wisconsin

Bare-metal execution: Utah and Wisconsin run checkpointable workloads under SLURM on 12 emulated locations.

Controlled WAN: Linux `tc/netem` impose realistic bandwidth, latency, and packet-loss conditions.

Google Cloud

Live deployment

Real deployment: 7 `e2-medium` VMs span 7 GCP zones.

Real conditions: Magellan experiences public Internet routing, provisioning variability, compute prices, and actual egress fees.

Checkpointable Workloads

LLM Training

Native PyTorch checkpoints preserve model and optimizer state.

Stresses: memory, I/O, checkpoint transfer, and dataset pre-staging.

Dendro-GR

Numerical-relativity workload using native solver checkpoints.

Stresses: sustained CPU execution, pauses, migration, and heterogeneous performance.

pyperformance

General CPU benchmark suite migrated transparently using CRIU.

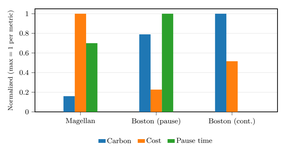
Stresses: process migration, per-node execution rates, and destination-capacity arbitration.

Evaluation and Results

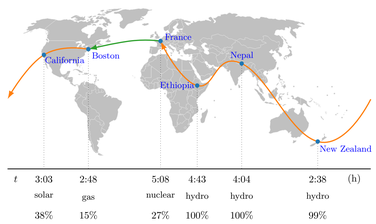
Metrics: carbon, cost, makespan, utilization

CloudLab — 13× Carbon Reduction

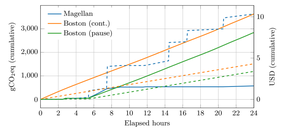
Normalized multi-objective tradeoffs



Optimal Path

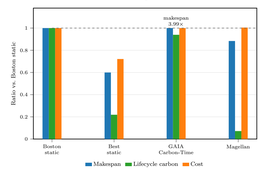


Carbon and cost accumulation

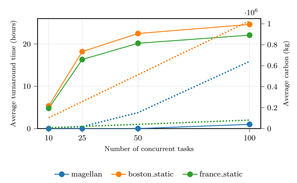


Google Cloud — Scales 100× concurrent tasks

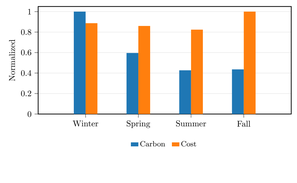
Normalized multi-objective tradeoffs



Concurrent task scaling



Seasonal variation



Magellan Models

$$x^* = \arg\min_{x \in \{\text{continue, pause, migrate}\}} (\alpha \widetilde{T}_x + \beta \widetilde{C}_x + \gamma \widetilde{\$}_x)$$

α, β, and γ encode the priorities assigned to makespan, carbon, and monetary cost.

Continue

$T_{\text{cont}} = T_c \quad C_{\text{cont}} = P\,T_c\,C_{\text{src}} \quad \$_{\text{cont}} = T_c\,P\,r_{\text{src}}$

Pause

$T_{\text{pause}} = t_p + t_i + t_r + T_c \quad \$_{\text{pause}} = T_c\,P\,r_{\text{src}} + t_i\,P\,r_{\text{src}}^{\text{idle}}$

$C_{\text{pause}} = P(t_p + t_r)C_{\text{src}} + P\,T_c\,\overline{C}_{\text{src}}(t_0 + t_i)$

Migrate

$T_{\text{mig}} = t_p + t_x + t_r + T_c' \quad \$_{\text{mig}} = T_c'\,P\,r_{\text{dst}} + S\ \text{TCGB}$

$C_{\text{mig}} = P(t_p + t_x)C_{\text{src}} + P(T_c' + t_r)\overline{C}_{\text{dst}} + C_{\text{transfer}}$

t_p, t_i, t_x, and t_r denote pause, idle, transfer, and restore time. Destination execution time is $T_c' = T_c(r_{\text{src}}/r_{\text{dst}})$. Network transfer carbon is $C_{\text{transfer}} = E_{\text{transfer}} C_{\text{net}}$, where $E_{\text{transfer}} = S(\epsilon_0 + \epsilon_d d/1000)$.

Short-horizon carbon forecast

Magellan fits a linear trend over the most recent carbon samples:

$$\hat{C}(\tau) = C(t_0) + m(\tau - t_0), \qquad \overline{\hat{C}}_{\text{dst}} = \frac{1}{t_w}\int_{t_0}^{t_0+t_w} \hat{C}(\tau)\,d\tau.$$

The window average estimates the carbon intensity the task will experience after arriving at a destination.

Migration guardrails

Migration is selected only when projected savings outweigh checkpoint, transfer, restore, and network-energy overheads. It is pruned whenever $K + \$_{\text{mig}} \geq K_{\max}$.

Findings and Future Work

Key Findings

- **Transient opportunities matter:** repeatedly exploiting temporarily clean regions is more robust than migrating once to a permanent "green haven."
- **Migration must be amortized:** under measured conditions, cross-region migration generally requires roughly **10–30 minutes** of remaining execution time to recover its overhead.
- **Capacity changes the opportunity:** Carbon savings fall from **57.9%** at low utilization to **9.9%** near cluster saturation, with no destination oversubscription.

Limitations and Next Steps

- Improve carbon & bandwidth forecasting
- Support heterogeneous CPU/GPU hardware, VRAM state, RDMA connections, and cross-architecture restoration.
- Explore WASM and container-native checkpointing to improve portability.
- Incorporate discounted cloud Spot nodes.



Explore Magellan

Paper • Code • Results

Takeaway. Decentralized stateful carbon migration is practical. Magellan reduces emissions by up to **13×** while achieving **11.6%** **lower makespan** at similar cost, and remains capacity-safe as competing workloads scale.

References

[1] Electricity Maps, "Real-Time Carbon Intensity: A Data-Driven Insight into the Future of Energy Systems," 2020.
[2] R. Ricci, E. Eide, and The CloudLab Team, "The CloudLab Testbed," *;login:*, 2014.
[3] Google LLC, "Google Cloud Platform," 2025.
[4] M. Fernando, D. Neilsen, E. Hirschmann, and H. Sundar, "A Scalable Framework for Adaptive Computational General Relativity on Heterogeneous Clusters," *ICS*, 2019.
[5] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," *EMNLP System Demonstrations*, 2020.