University of Massachusetts Amherst BE REVOLUTIONARY™

Ecovisor

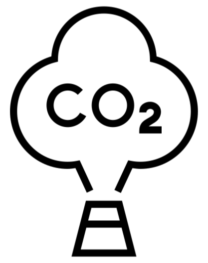
A Virtual Energy System for Carbon-Efficient Applications

Abel Souza, Noman Bashir, Jorge Murillo,
Walid Hanafy, Qianlin Liang, David Irwin, Prashant Shenoy

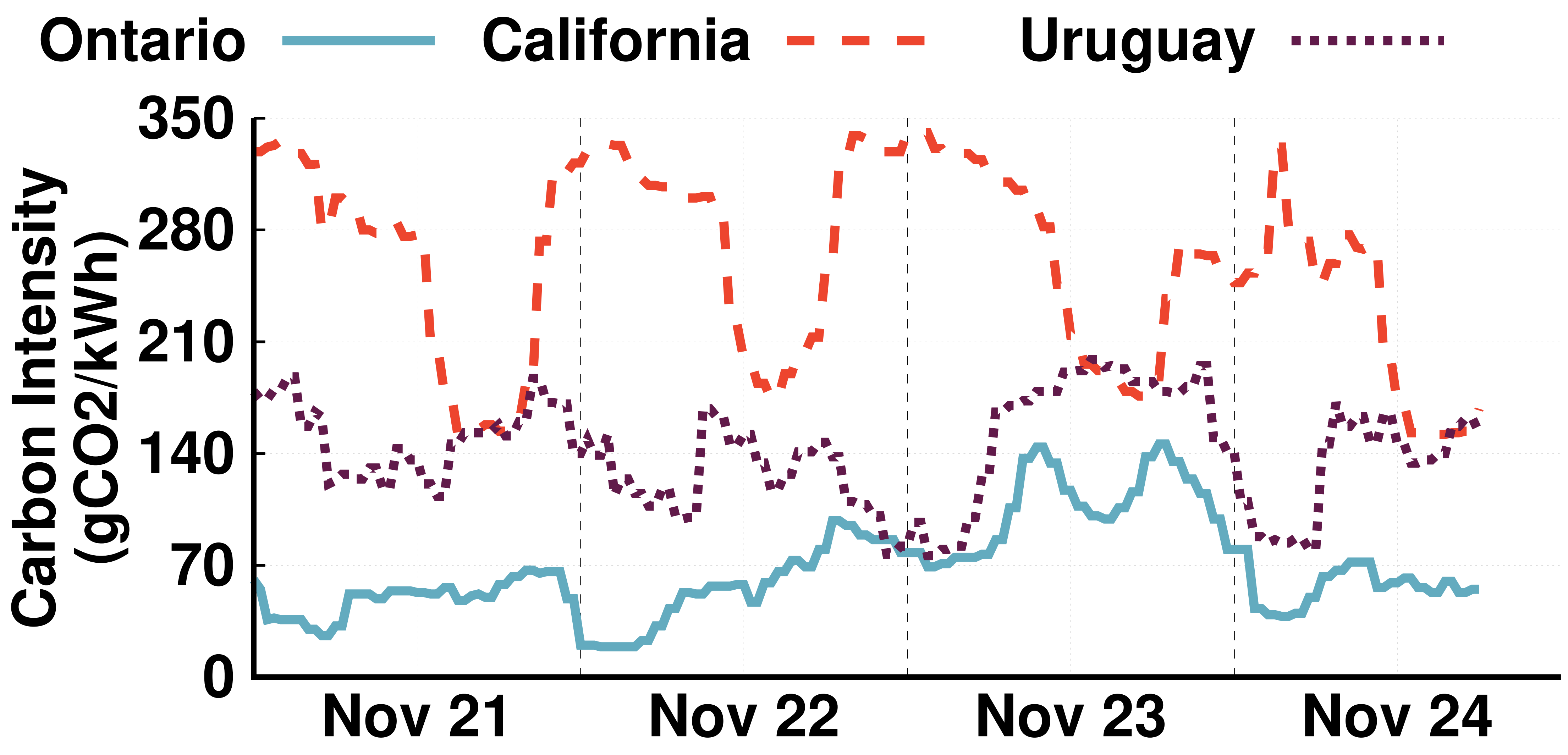
The Sustainability Problem

- Cloud capacity[1] — and energy use — doubling every ~4-5 years
- Rising energy usage is not *really* the problem
- We must (eventually) **reduce emissions to ~0**
- Shift focus from energy to **carbon:**

`Carbon-efficiency != Energy-efficiency`



`= Energy x Carbon_Intensity`



Carbon intensity* can vary from **less than 50g** to **more than 800g** across time and geographical regions.

Source: electricityMap.org

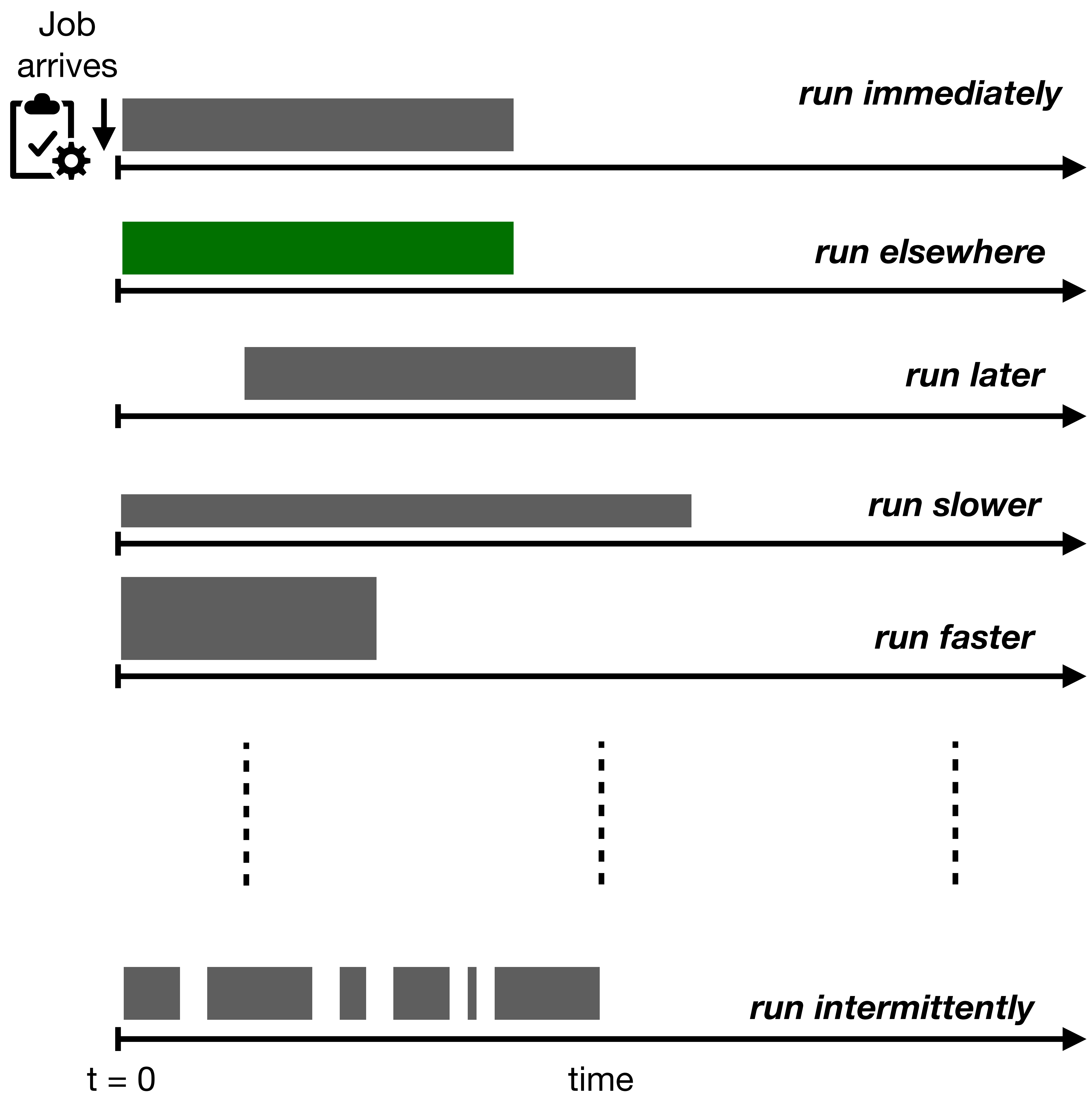
Clean Energy is Unreliable, and varies widely both temporally and geographically.

[1] https://www.srgresearch.com/articles/hyperscale-data-center-count-reaches-541-mid-2020-another-176-pipeline

Computing's Unique Advantages

- Modern workloads have key temporal and spatial execution flexibility:
  - Deadline / Slack
  - Migration / Load-balancing
  - Autoscaling / Frequency scaling
  - Suspend (Checkpoint) & Resume (Restart)

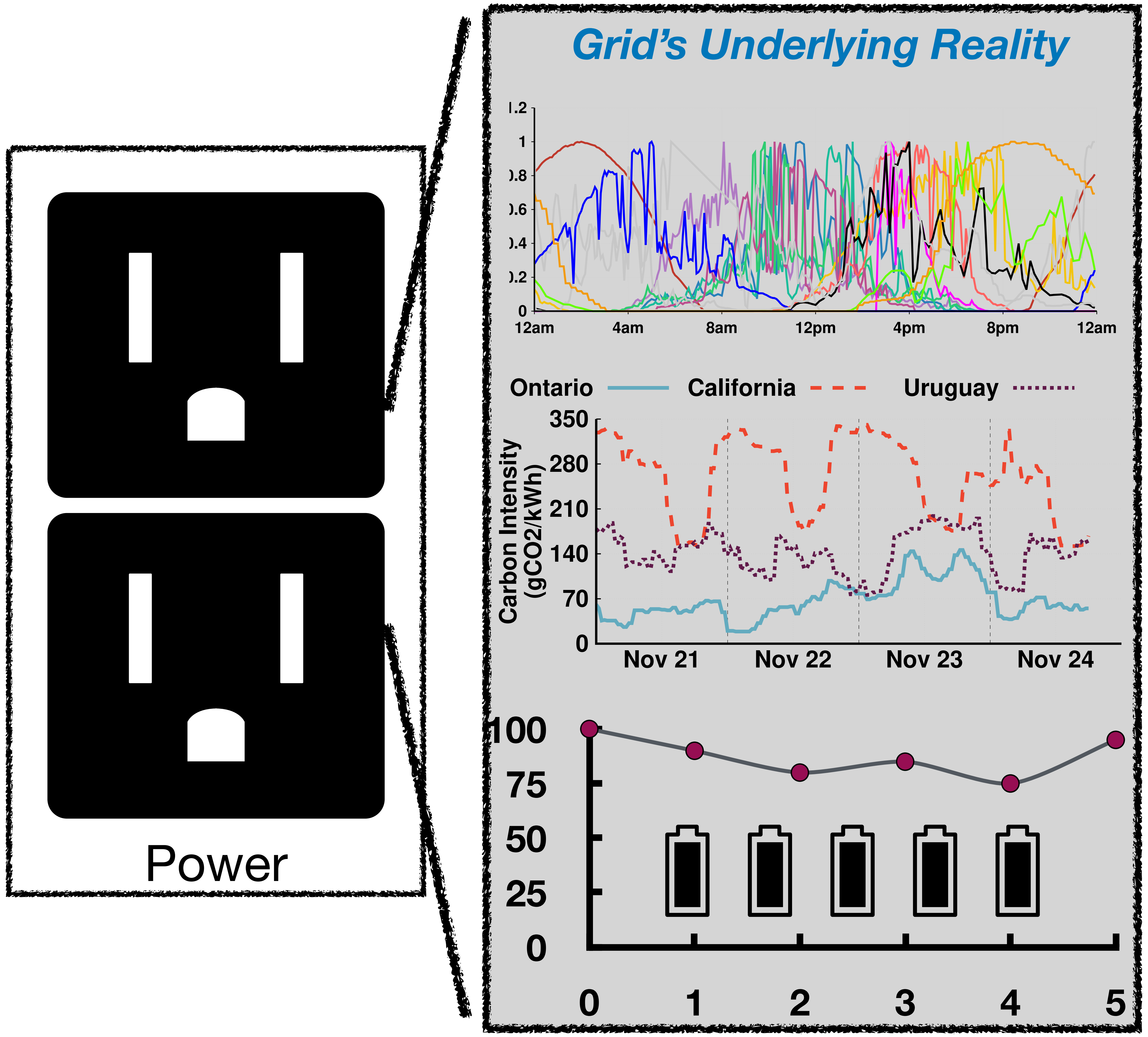
Today's applications can use various techniques to regulate power consumption and handle the fluctuation and unpredictability of renewable energy.



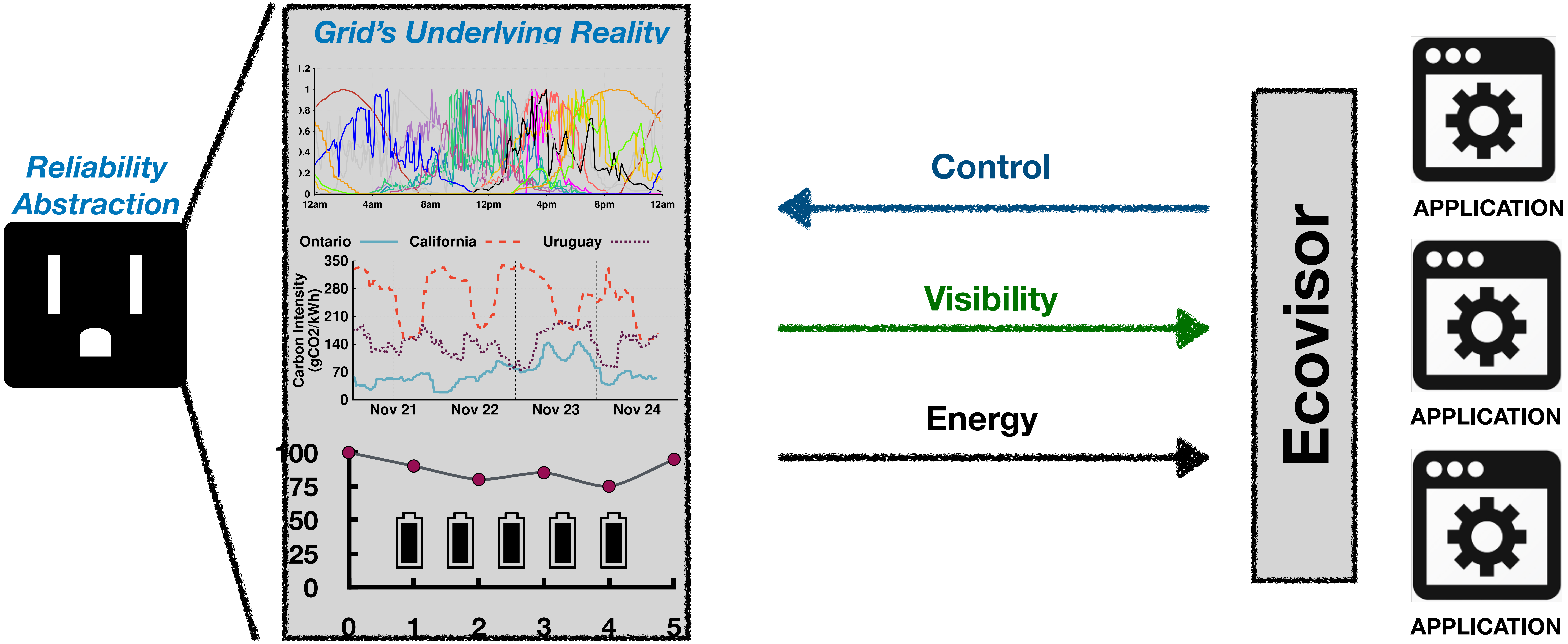
Issue: Energy’s Reliability Abstraction Limits Computing’s Potential

- Today's energy systems mask clean energy’s unreliability from applications in hardware
- The only abstraction is a reliable supply of power on demand:
  - Devices, including servers, via their electrical **socket interface**
- Energy system now includes a connection to not only the grid:
  - Energy storage, e.g., batteries
  - Access to solar/wind

Without visibility into the Grid’s reality, applications cannot influence their own energy and compute demand.



Solution: Expose visibility and software-defined control of the energy system

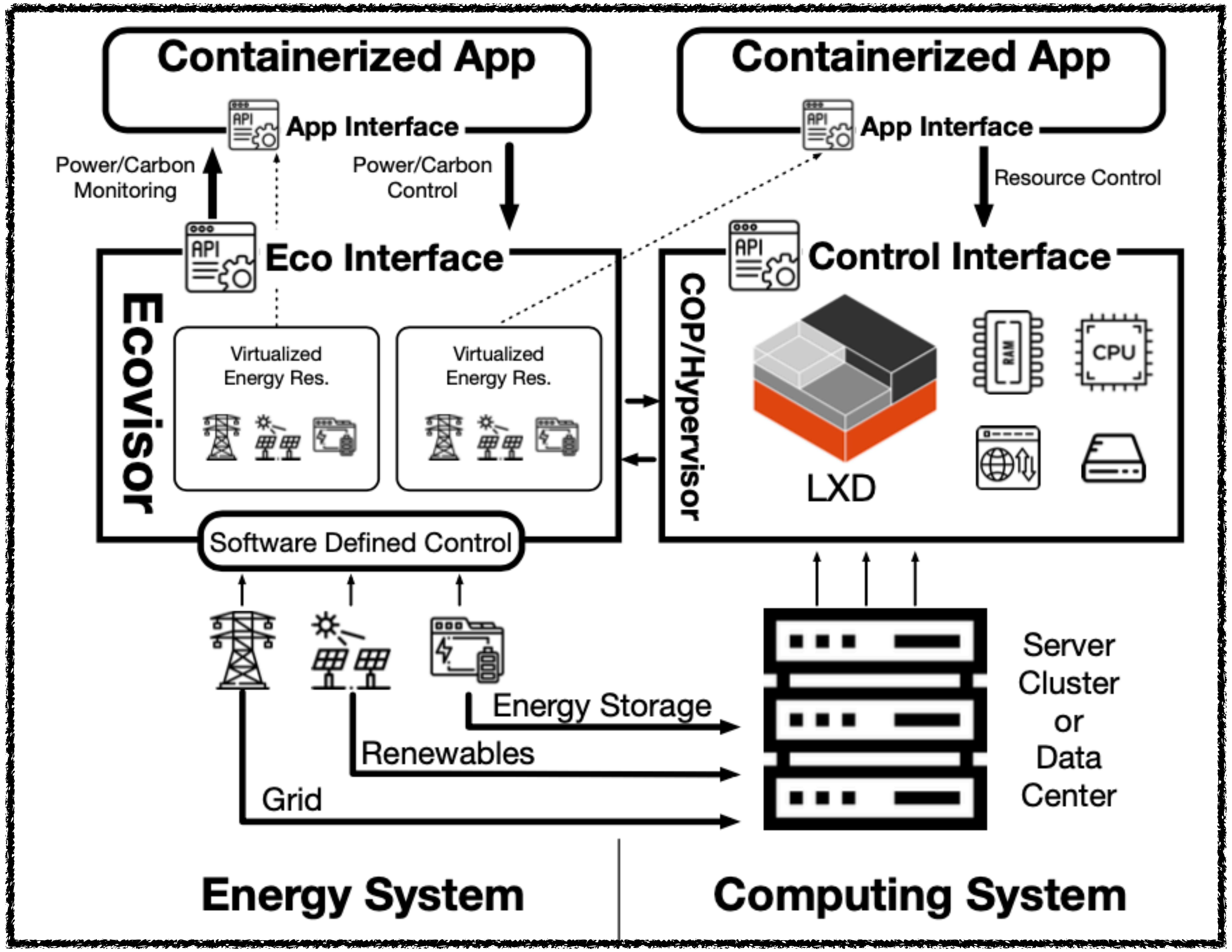


The visibility and control of the energy system allows applications to monitor energy usage, track emissions, and select renewable sources — e.g., solar — through software control knobs.

Ecovisor

Virtualize the energy system to enable the design of carbon-efficient applications

- Real-time energy information to applications through an API
  - **Local** and **Cluster** support
- Applications implement their own energy policies
  - Energy usage enforced through control groups (`cgroups`)
  - Container-level API (with VM support)
  - Software-defined power meters with PowerAPI

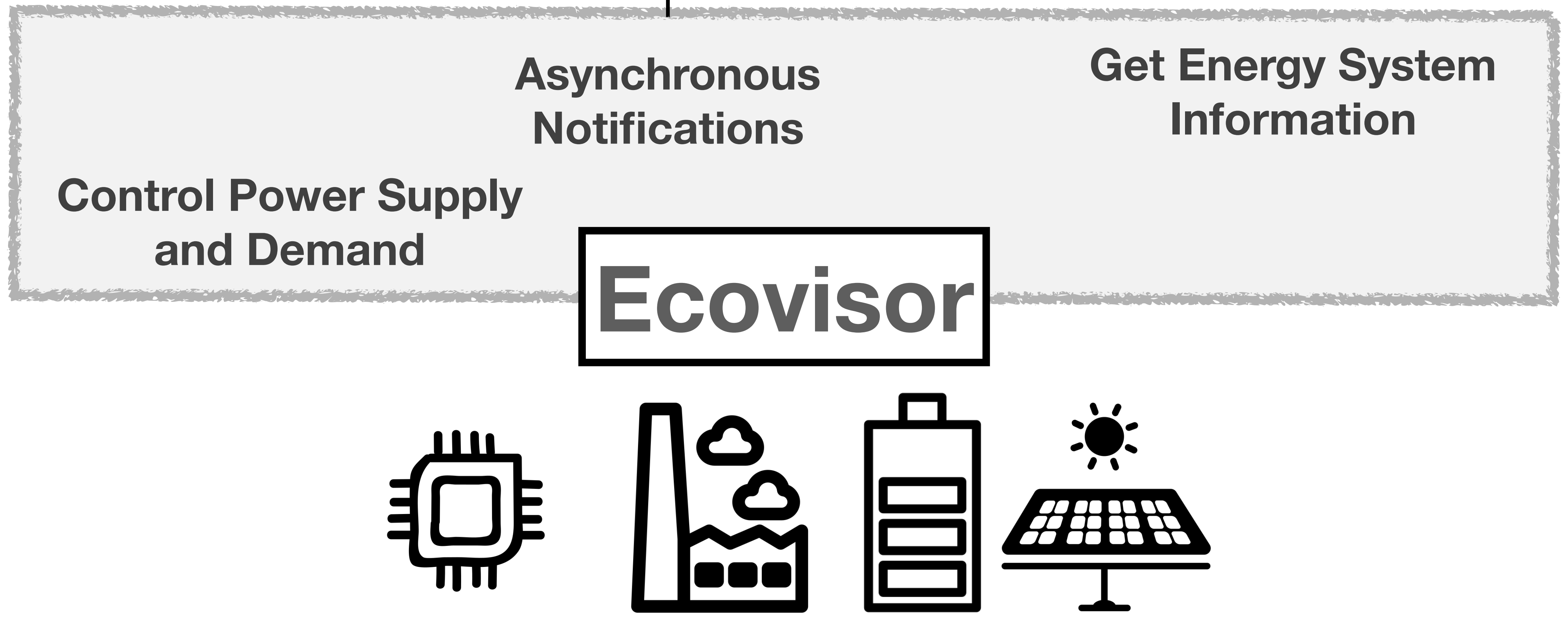


The Ecovisor exposes a *Simple* and *Narrow* API by Design

- Functions to **`control()`, `get()`**, and **`notify()`** events about the *physical* power's supply and demand
- This API enables wide range of policies:
  - Higher-level library-interfaces and abstractions
  - Simplify interactions with the virtual energy system
  - Entirely transparent to applications

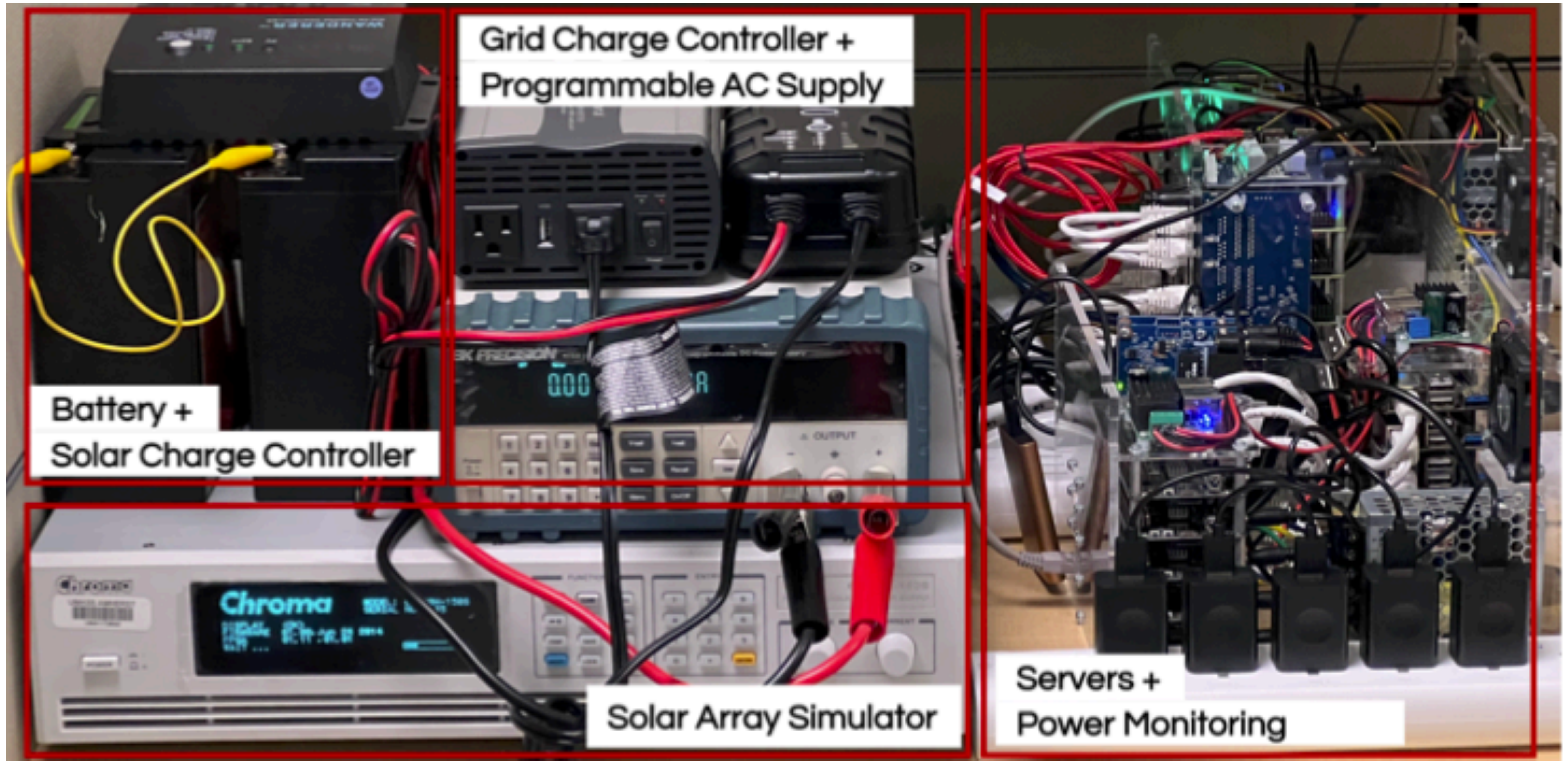
An example of a Library API

Function Name	Description
`get_container_energy()`	Energy usage in interval (t_1, t_2)
`get_container_carbon()`	Carbon usage in interval (t_1, t_2)
`get_app_power()`	Power usage for an application
`get_app_energy()`	Energy usage in interval (t_1, t_2)
`get_app_carbon()`	Carbon usage for an application
`set_carbon_rate()`	Set carbon rate for a container
`set_carbon_budget()`	Set carbon budget for a container
`set_app_carbon_budget()`	Set application's carbon budget
`notify_solar_change()`	Called when solar changes
`notify_carbon_change()`	Called when grid carbon changes
`notify_battery_full()`	Called when battery fully charged
`notify_battery_empty()`	Called when battery empty



Ecovisor: Prototype Implementation - Some results

- Software: REST API
  - Access to energy APIs and CO2signal
  - Generic Interface
    - Can connect to any container system
- Hardware: small-scale prototype



- Reducing carbon (ML training, MPI)
  - System (WaitAWhile - Middleware'21)
  - App-specific (Wait&Scale)
- Budgeting carbon (web server)
  - System (rate limiting)
  - App-specific (budgeting)

Require: target_carbon_rate

for each t= 1,2,... **do**

  $app\_power = \texttt{get\_app\_power(t)}$

  $grid\_carbon = \texttt{get\_grid\_carbon(t)}$

  $app\_carbon = app\_power \times grid\_carbon$

  $delta\_carbon = app\_carbon - target\_carbon\_rate$

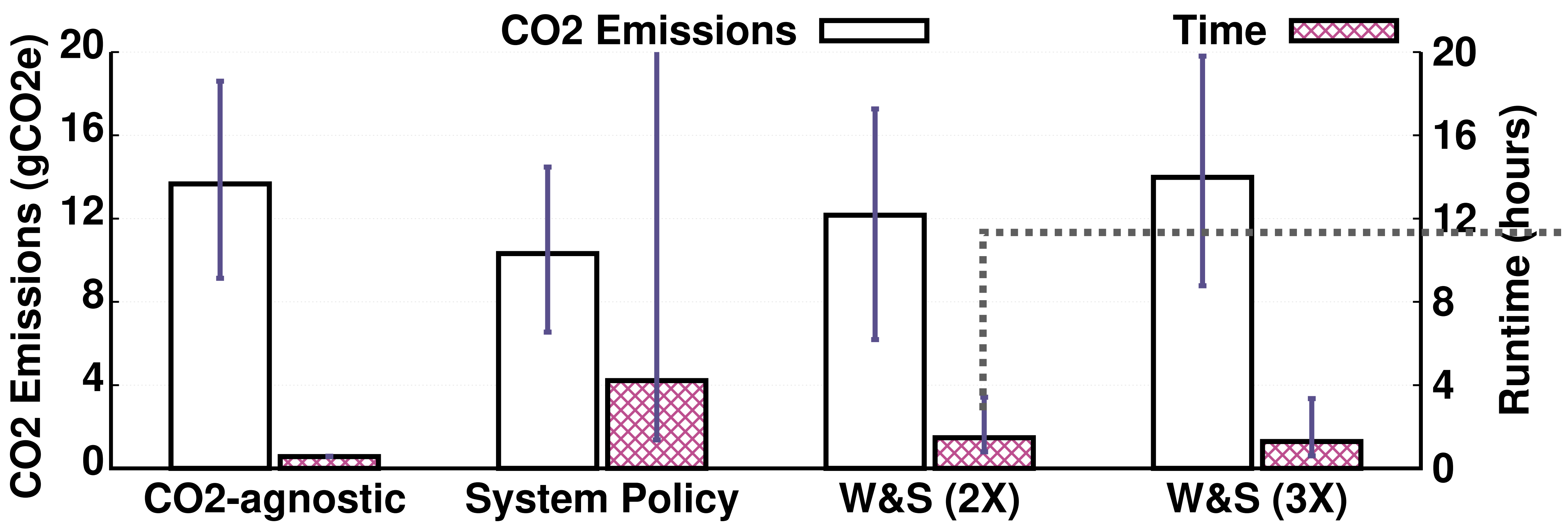
  $delta\_containers = int(delta\_carbon/ \texttt{get\_container\_carbon()})$

  $containers(t) = containers(t-1) + delta\_containers$

end for

Ecovisor: Optimizing Carbon/Performance Trade-off

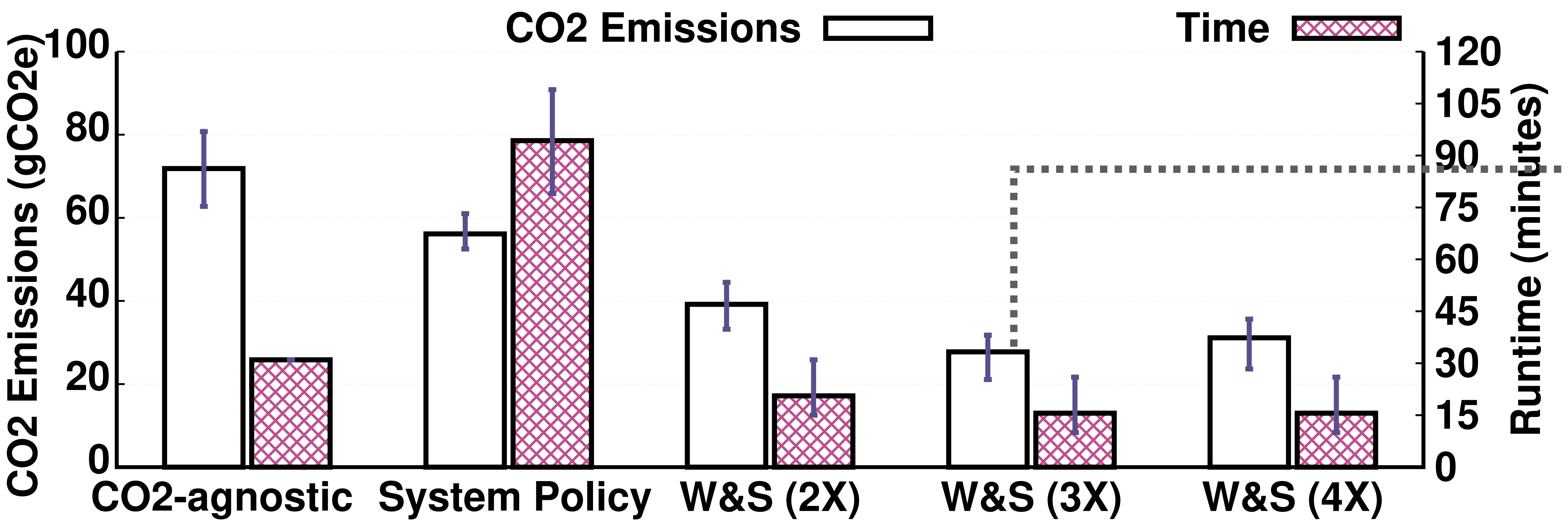
- System (WaitAWhile - [1]) versus Application-specific (Wait&Scale) policy



PyTorch ML Training

Optimal Scale = 2X

Under-review work on leveraging workload elasticity.



BLAST

Optimal Scale = 3X

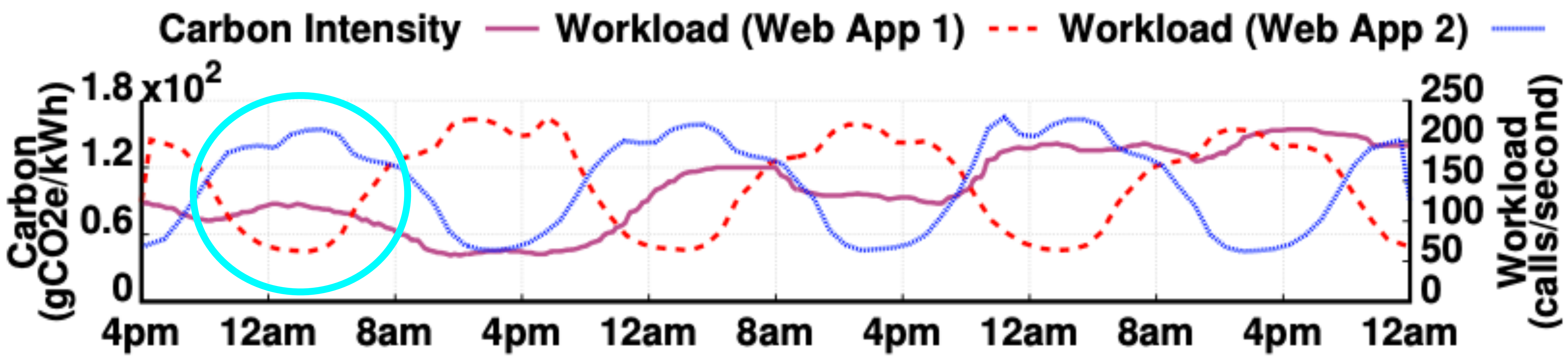
Embarrassingly parallel HPC job.

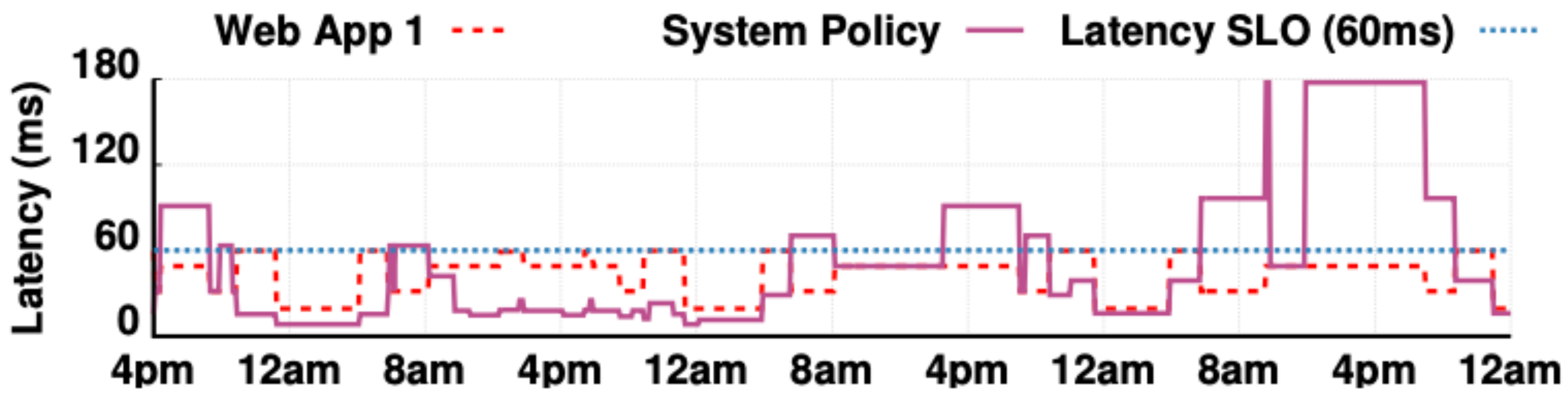
Allow applications to better optimize their carbon-efficiency compared to a system-level policy (application-agnostic)

[1] Wiesner, Philipp, et al. "Let's wait awhile: how temporal workload shifting can reduce carbon emissions in the cloud." *Proceedings of the 22nd International Middleware Conference*. 2021

Ecovisor: Carbon Budgeting

- System (**carbon rate-limiting**) versus Application-specific (carbon budgeting) policy





Key Point: ***Application-specific carbon budgeting provides useful flexibility: ~23% overall gains on top of the system-level policy.***

Conclusion

- **Key Point:** Many carbon-efficiency optimizations possible if applications have visibility/control
    - Applications are better positioned to take their own decisions
    - However, libraries using the Ecovisor can offload this task from applications

- **Ecovisor exposes** useful functions to **enable carbon-efficient** applications
    - Access to energy APIs and CO2signal

- A **Foundation** for developing new abstractions to simplify **developing carbon-efficient applications**.

- **Ongoing Work:** Exploiting flexibility to reduce carbon; developing new abstractions for Ecovisor

Thank You!

asouza@cs.umass.edu

Ecovisor

A Virtual Energy System for Carbon-Efficient Applications

Abel Souza, Noman Bashir, Jorge Murillo, Walid Hanafy, Qianlin Liang, David Irwin, Prashant Shenoy