

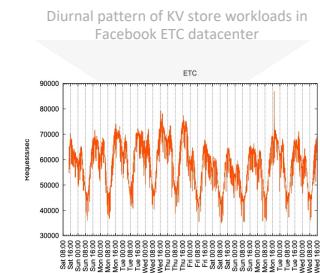


Peafowl: In-application CPU Scheduling to Reduce Power Consumption of In-memory Key-Value Stores

Showan Esmail Asyabi, Azer Bestavros (Boston University), Erfan Sharafzadeh (Johns Hopkins University), Timothy Zhu (The Pennsylvania State University)

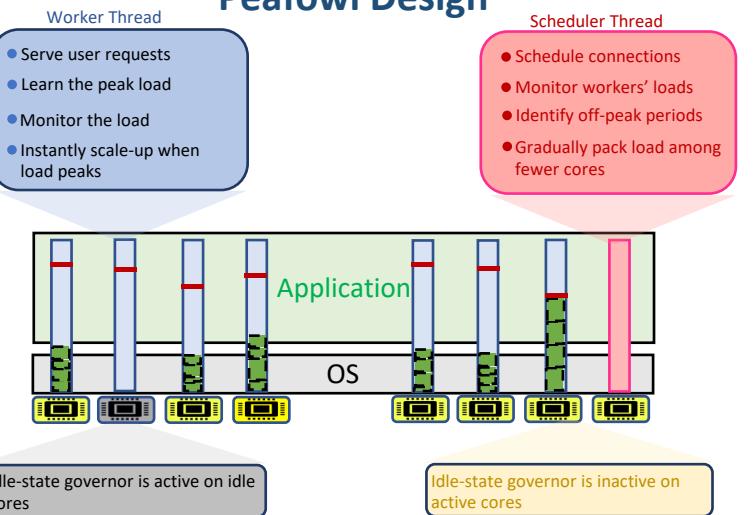
Problem and Motivation

- Tail latency of key-value (KV) stores impacts the overall performance of high-fanout applications
- KV store workloads have a diurnal pattern
- Service providers provide for peak load to ensure low tail latency
- Problem:** High energy consumption of ever-growing in-memory KV stores (i.e., cache nodes) in data centers



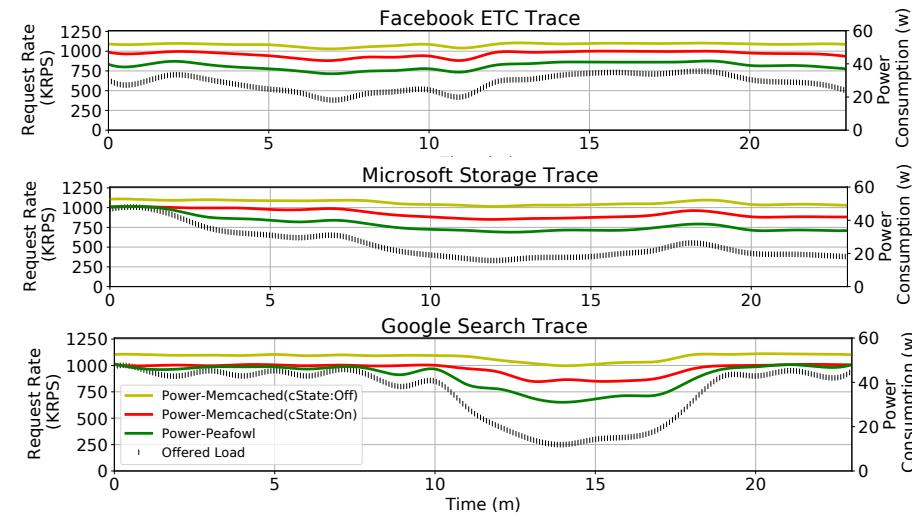
Goal: Save power during off-peak periods while ensuring microsecond scale tail latency

Peafowl Design



Idea: Perform scheduling in the KV store to unbalance the load during off-peak periods

Peafowl in Action

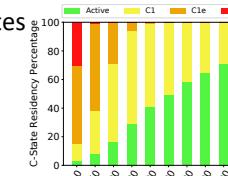


Peafowl saves 36% more power while keeping tail latency at microsecond scale

Existing Solutions

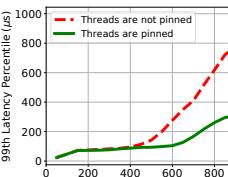
Idle-state governor: Force CPU into deep idle states

Problem: Short interarrival fragments idle periods



Feedback-based controllers: Monitor the load and adjust the number of allocated cores

Problem: Controllers rely on OS for scheduling → too slow

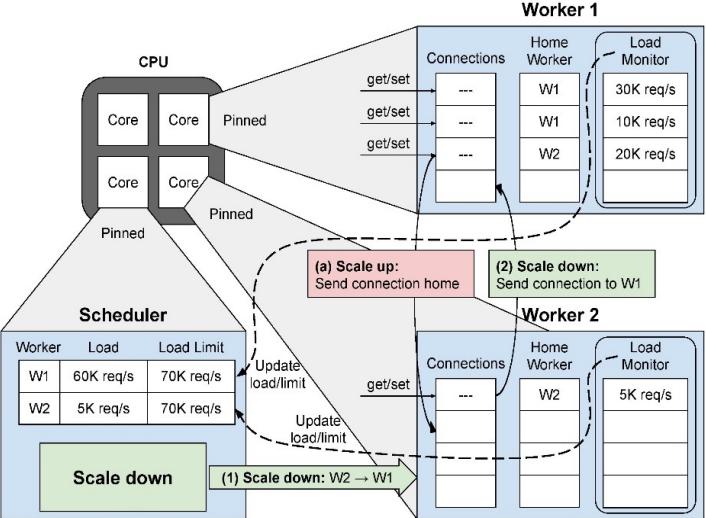


DVFS and request delaying: Exploit the latency gap to slow down the request processing

Problem: Due to the high arrival rate and short service time of KV store workloads, these approaches are not able to notably save power

The high arrival rate, short service time, and tight latency requirements make existing solutions less effective

Peafowl Implementation

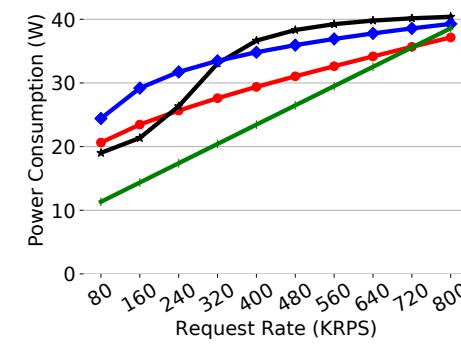


Open sourced at <https://github.com/showanasyabi/peafowl-kvs>

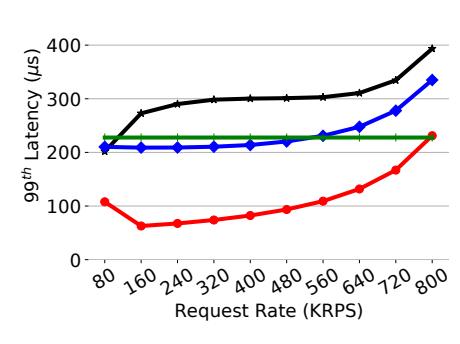
Peafowl Compared to Existing Approaches

Legend: CISG (red circle), Rubik (black line), uDPM (blue diamond), Peafowl (green line)

Target Latency = 300μs



Target Latency = 300μs



Peafowl outperforms Rubik, μDPM, and a Clairvoyant idle-state governor with up to 40%, 54%, and 45% more power savings respectively