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

**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 and 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

**Peafowl Implementation**
- Worker Thread: Serve user requests, learn the peak load, monitor the load, instantly scale-up when load peaks
- Scheduler Thread: Schedule connections, monitor workers’ loads, identify off-peak periods, gradually pack load among fewer cores

**Peafowl in Action**
- Facebook ETC Trace
- Microsoft Storage Trace
- Google Search Trace

**Peafowl Compared to Existing Approaches**
- Target Latency = 300μs
- Peafowl outperforms Rubik, µDPM, and a Clairvoyant idle-state governor with up to 40%, 54%, and 65% more power savings respectively

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

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

Open sourced at: https://github.com/showanasyabi/peafowl