Towards Power Management for FreeBSD

Robin Randhawa

robin.randhawa@arm.com

FreeBSD Developer Summit

Computer Laboratory

University of Cambridge

August 2015

Agenda

- An overview of Energy Aware Scheduling (EAS) for Linux
- Discussion on possibilities for FreeBSD

Motivations for EAS

Hardware topologies are becoming more varied, accommodating different power/ performance budgets

- SMP, multi-cluster SMP, ARM big.LITTLE technology
- Per core/per cluster DVFS (Dynamic Voltage & Frequency Scaling)

Linux power management frameworks are uncoordinated and hard to tune for different topologies

- cpufreq *vs* cpuidle *vs* scheduler

The Task Scheduler is best placed to orchestrate power-performance control

Conventional Scheduling

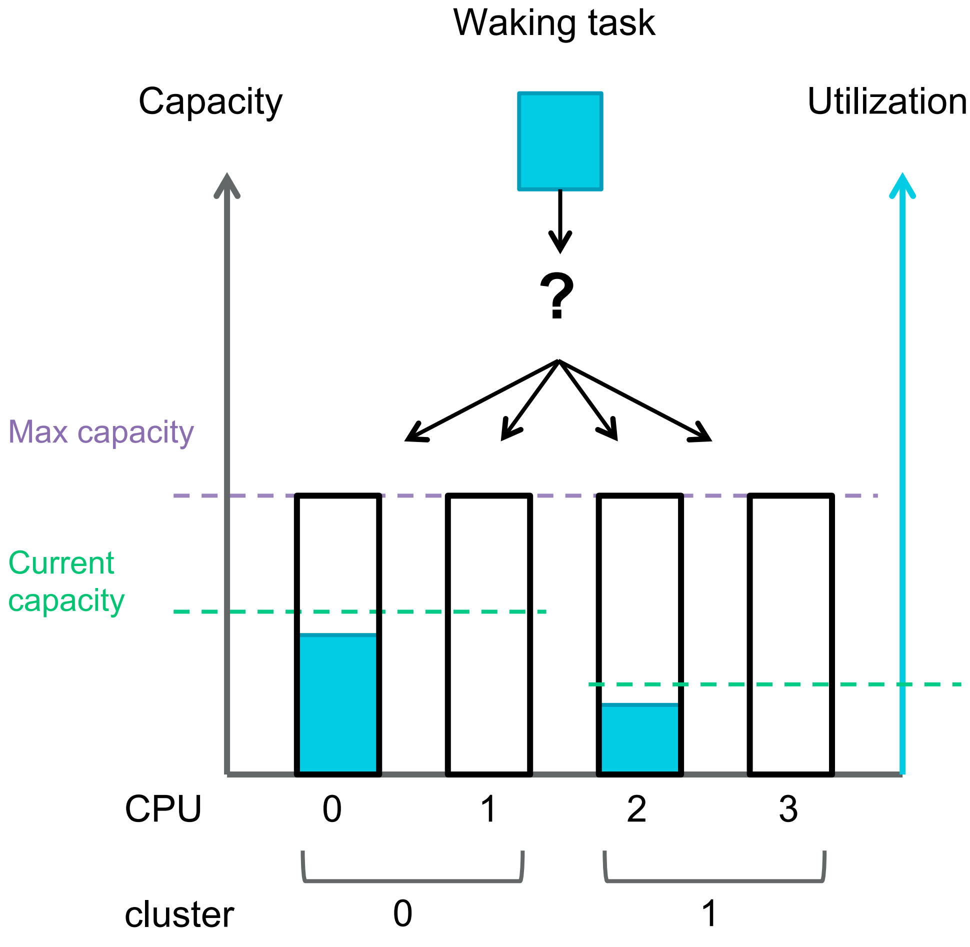
Scheduling policy decides task placement

- Affects performance and energy consumption

Mainline Linux policy is 'work preserving'

- Only cares about maximizing throughput
- DVFS and idle-states controlled by independent policy governors.

Designed for SMP, not energy-aware



Energy Aware Scheduling

Energy-Aware Scheduling (EAS) policy

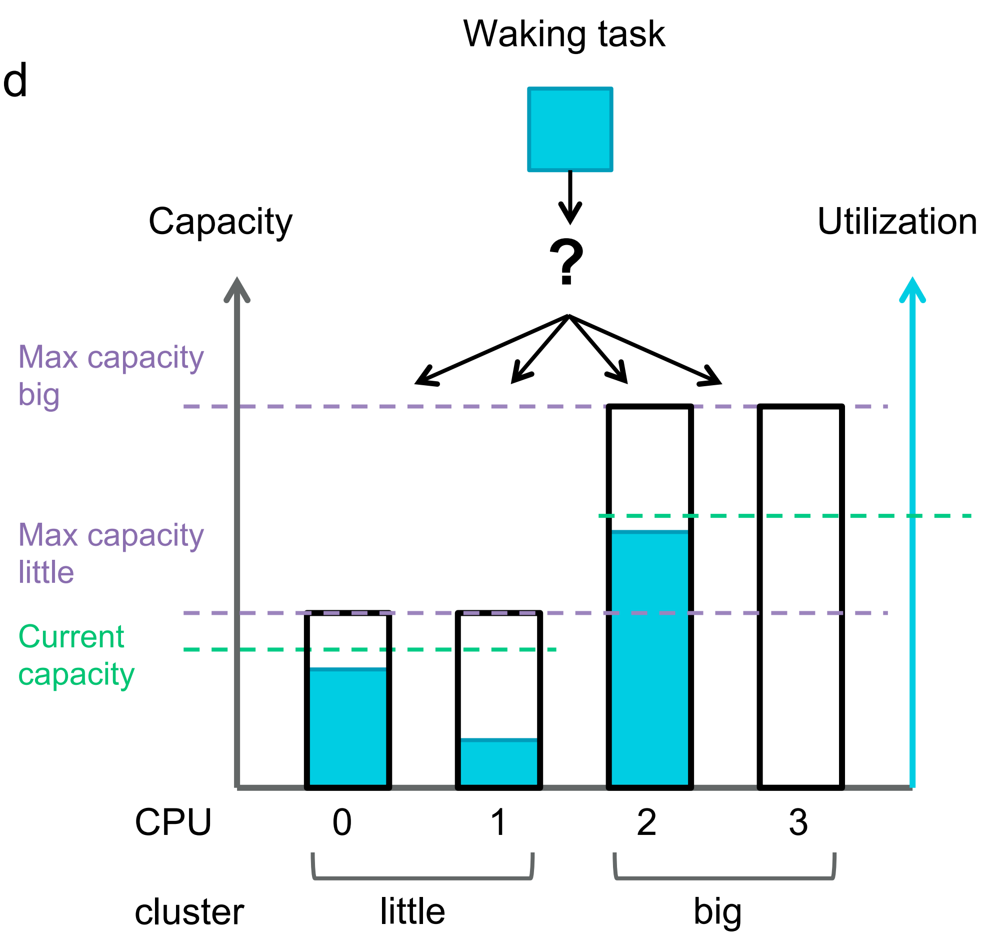
- Pick CPU with sufficient spare capacity and smallest energy impact

Requirements

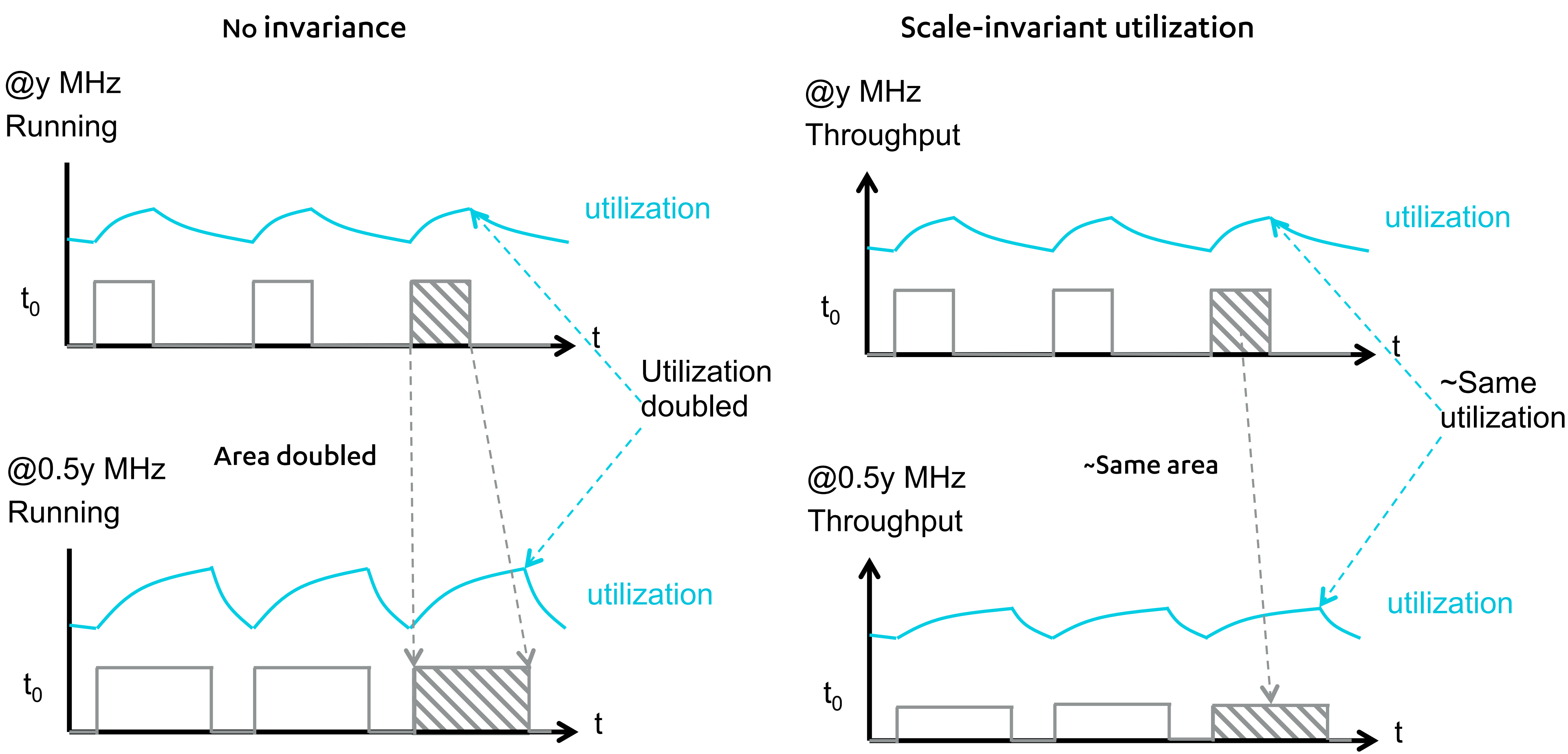
- Tracking of task utilization
- Platform energy model

Supports all topologies

- SMP
- big.LITTLE
- Async DVFS

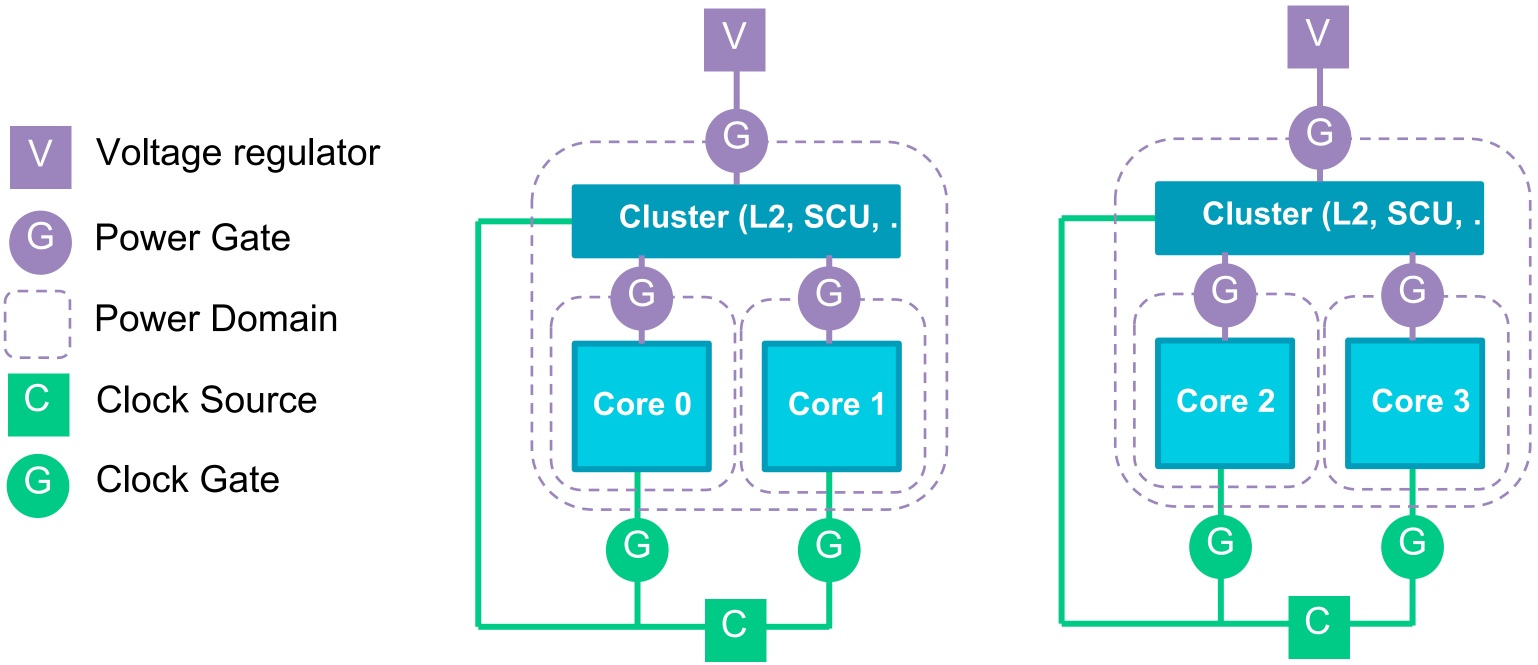


Scale invariant load



Component energy model

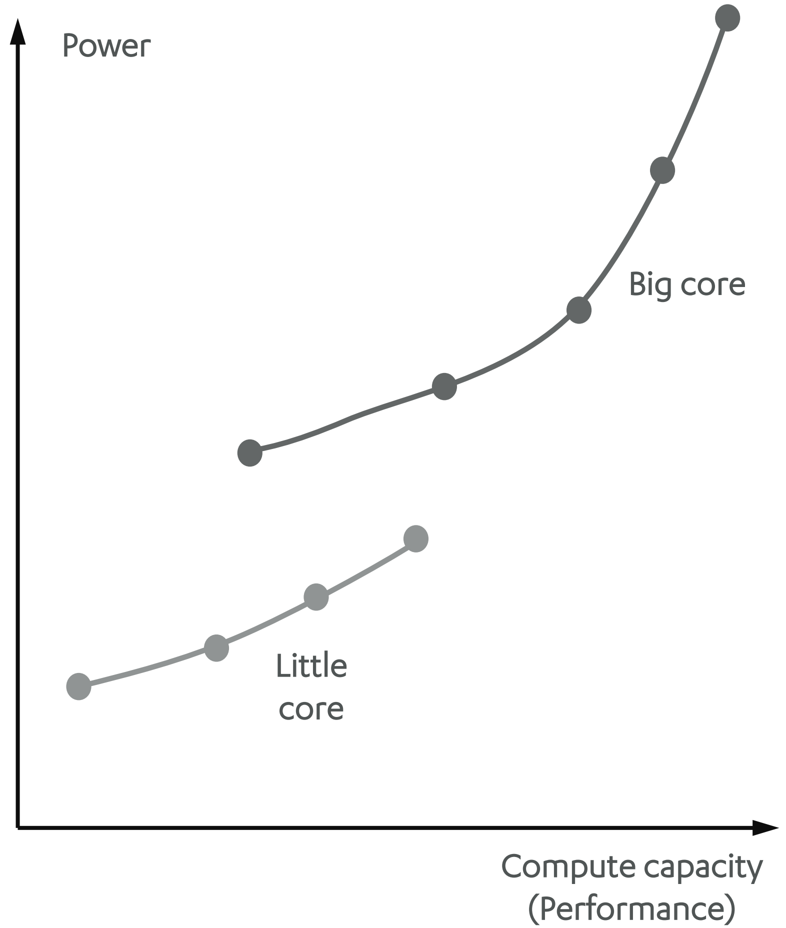
Tabular cost data for all power and frequency domains in the system



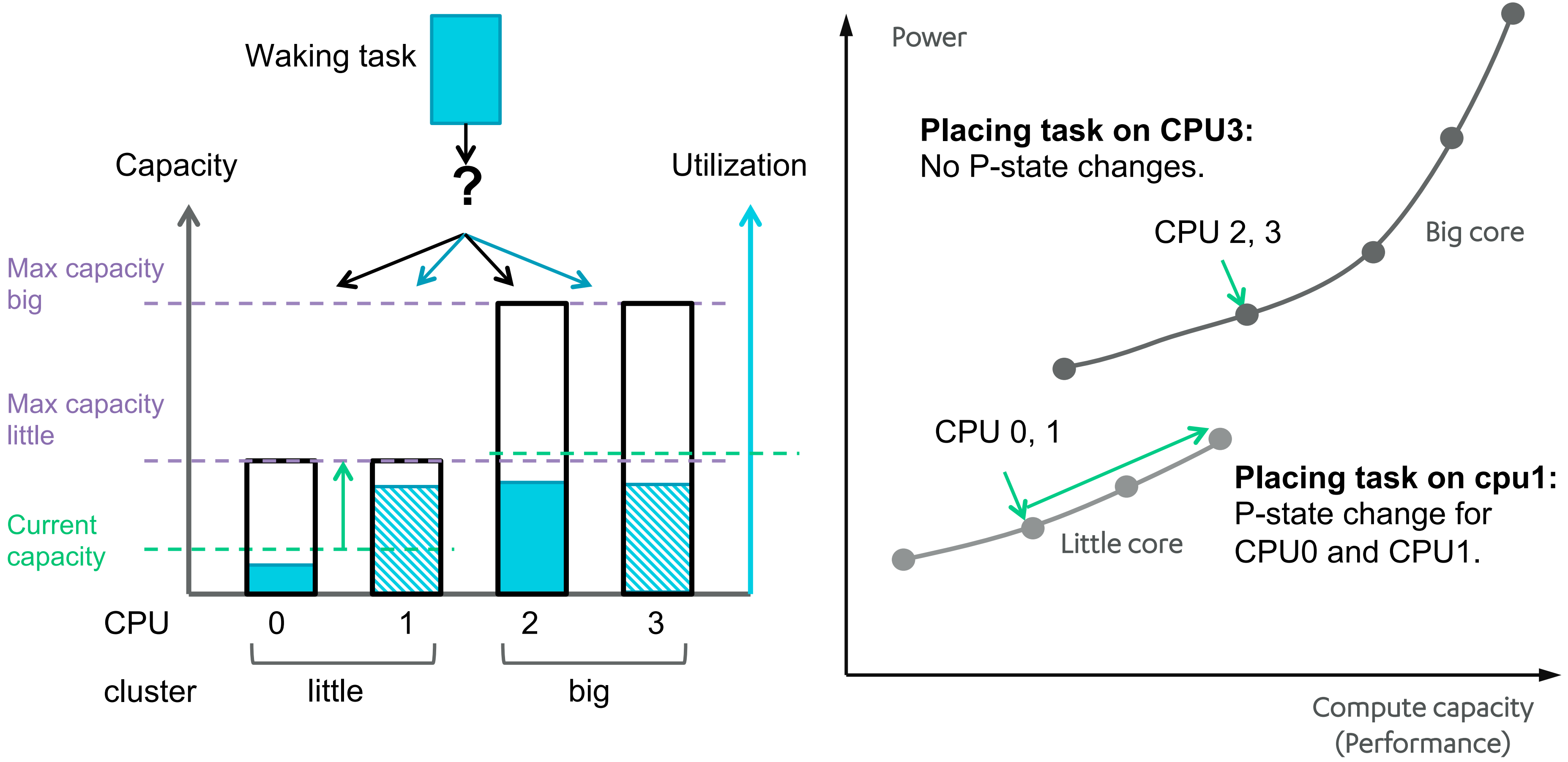
Energy model data

P-States (frequencies)	
Compute capacity	**Busy power**
Performance score normalised to the highest P-state of the fastest CPU in the system (1024)	Normalised power score (W)

C-States (Idle states)	
Idle power (normalised)	Normalised power score (W)

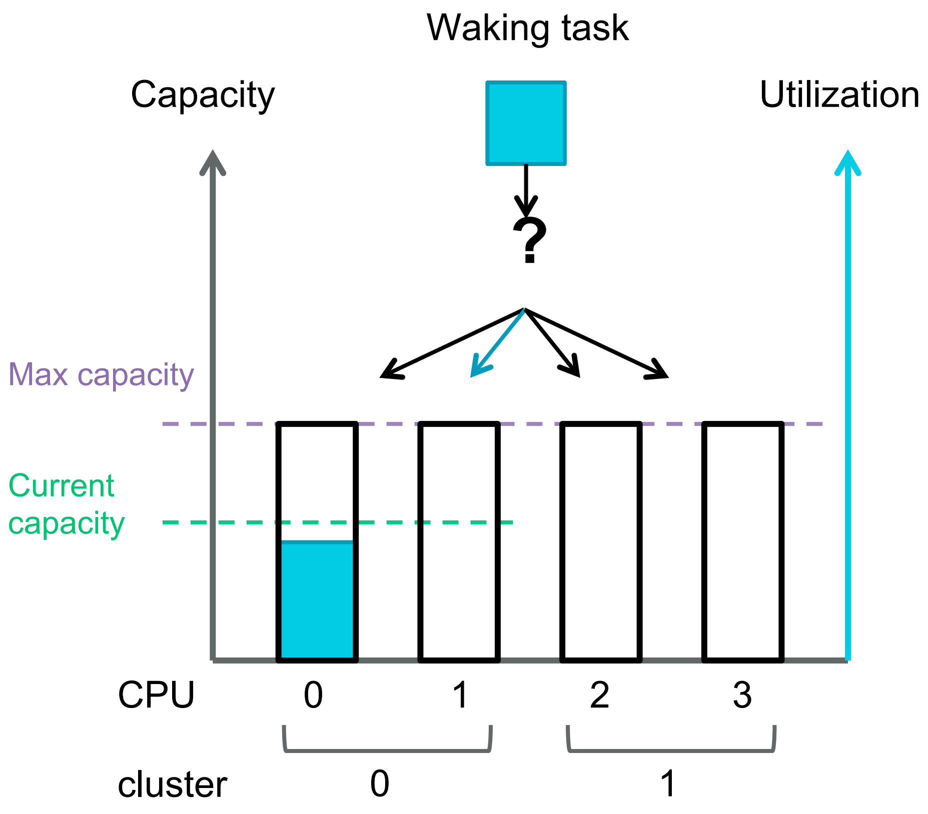


Estimating the energy impact



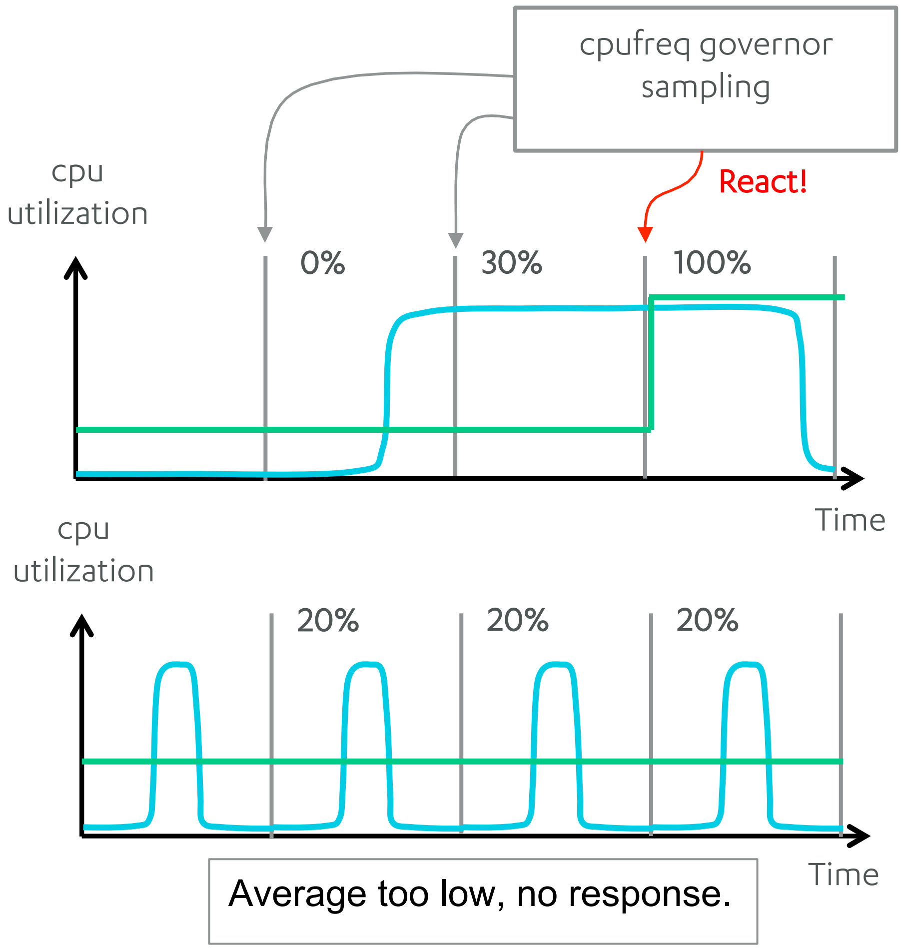
Idle state awareness

- Integration of cpuidle with the scheduler improves task placement on idle CPUs
- Scheduler picks CPU in shallowest idle-state (cheapest from a power and performance standpoint)



Conventional DVFS

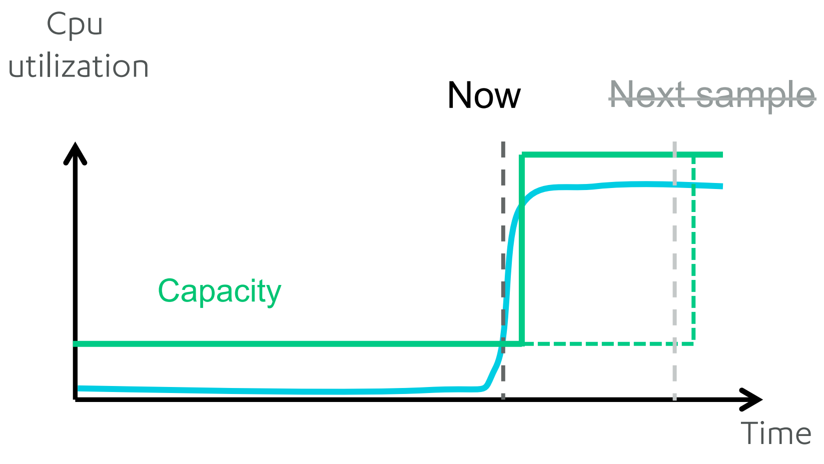
- Sampling based governors are slow to respond and hard to tune
- Sampling too fast: OPP* changes for small utilization spikes
- Sampling too slow: Sudden burst of utilization might not get the necessary OPP change in time.

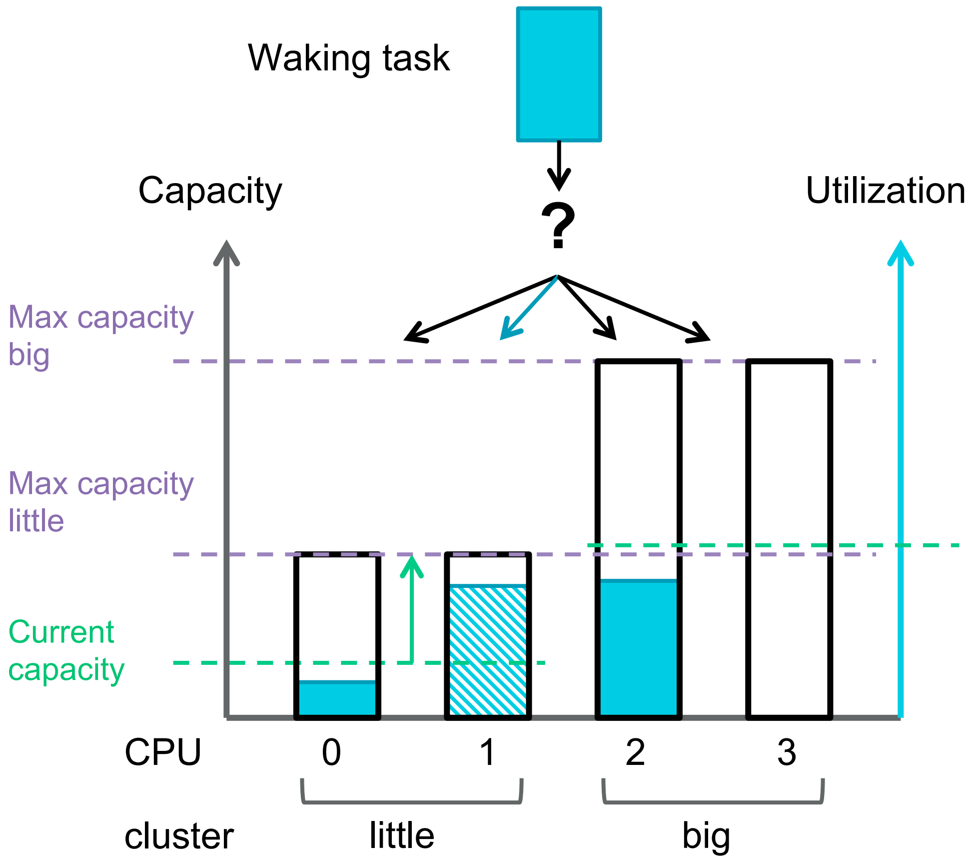


*OPP: Operating performance point (Voltage, frequency) tuple

Scheduler driven DVFS

- With scheduler task utilization tracking DVFS can be notified immediately when CPU utilization changes
- **Improved responsiveness**.





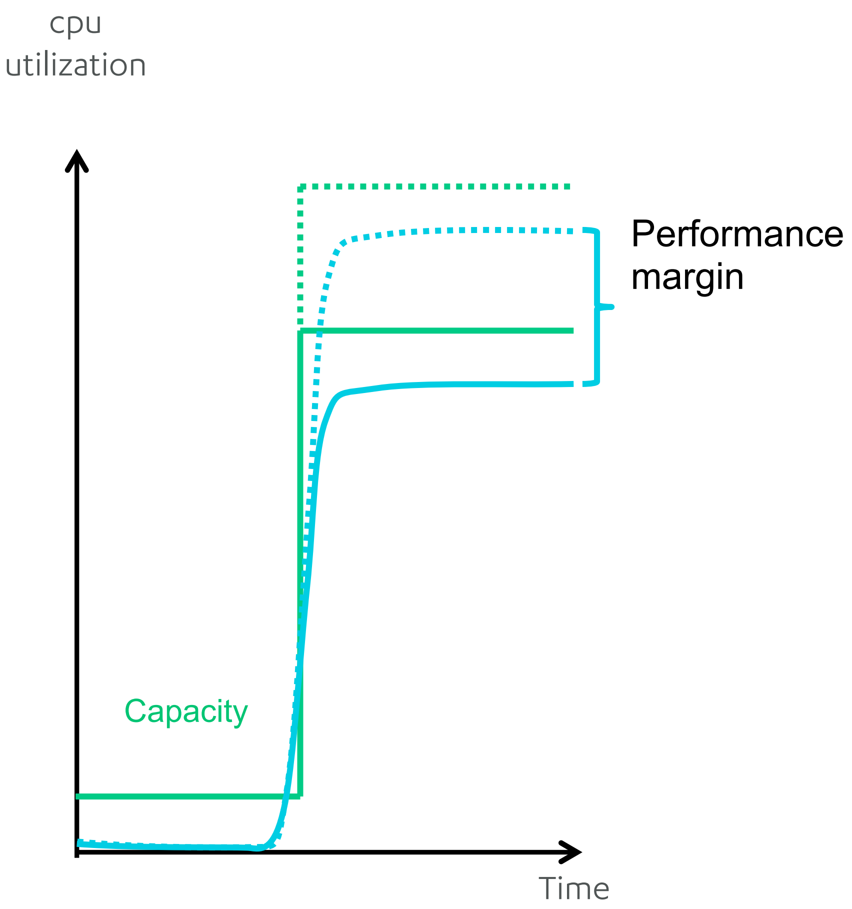
Centralised tunability

- **Current**: A set of governor-specific tunables.
- **Goal**: Single tunable to bias the energy/ performance trade-off.

Prototypes

Global boost tunable:
/proc/sys/kernel/sched_cfs_boost

Task group (cgroup) based tuning:
/sys/fs/cgroup/stune/<group>/ schedtune.boost



EAS related tools/utilities

rt-app – synthetic workload generator for Linux
https://github.com/scheduler-tools/rt-app

ARM “Workload Automation” – runs Android/ChromeOS tests
https://github.com/ARM-software/workload-automation

Kernelshark – trace analysis

https://git.kernel.org/pub/scm/linux/kernel/git/rostedt/trace-cmd.git

Improved analysis tools (experimental)

TRAPpy: Trace Analysis and Plotting in Python

BART: Behaviour Analysis and Regression Toolkit

https://github.com/ARM-software

Where do we go from here ?

(Very) early thoughts on incremental steps for FreeBSD

- Incremental steps
  - Basic
    - Settle on a suitable platform
      - Good documentation for power-perf machinery
      - Measurement capability
        - DAQs etc
        - Has to be CPU subsystem specific
    - Abstract task load scale
      - Modular load calc
        - runqueue residency time
        - micro-arch hints
    - Abstract capacity scale for processing elements
      - SMP
      - big.LITTLE
    - PE topology expression
    - Energy model expression
      - Modular architecture
    - Simple DVFS support
      - Modular policy
      - Driven from the scheduler
    - Simple cpuidle support
      - Scheduler aware
      - Task wakeup
    - Centralised tunability
      - At one end: Energy efficient operation at best possible throughput
      - At the other end: Outright performance at the intentional expense of energy
    - Tooling
      - Scheduling visualisation
      - DVFS and Idle visualisation
      - Energy model build flows
  - Advanced
    - Device tie-in
      - I/O subsystem driven constraint specification for CPU power management
      - Justin Hibbit’s clock control
    - Thermal
      - Thermal sense driven constraint specification for task scheduling and CPU power management
      - Energy model driven

End