Unifying Power Policies

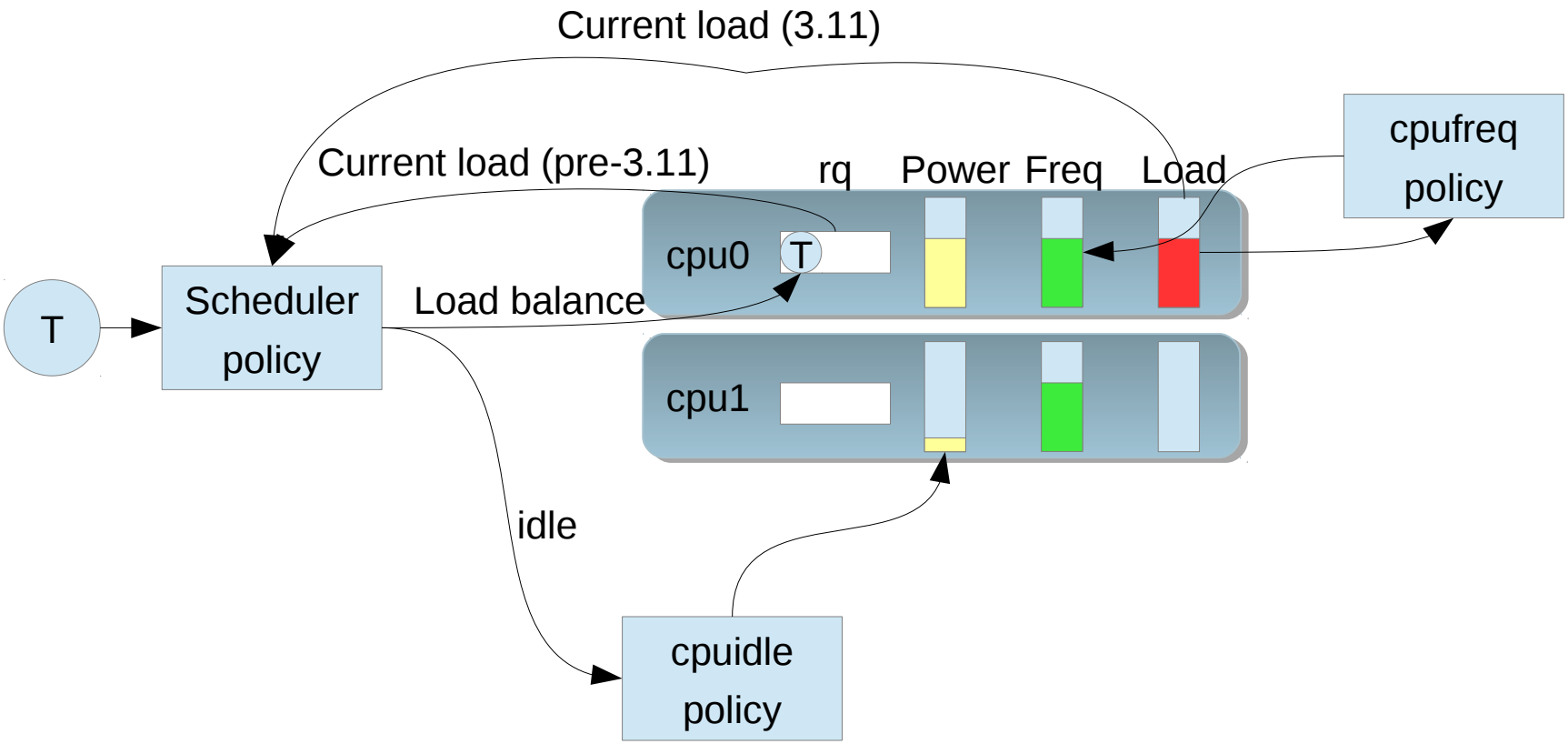
Linux Plumbers Conference 2013

Morten Rasmussen

Existing Power Policies

- Frequency scaling: cpufreq
  - Generic governor + platform specific driver
  - Decides target frequency based on overall cpu load.
- Idle state selection: cpuidle
  - Generic governor + platform specific driver
  - Attempts to predict idle time when cpus enter idle.
- Scheduler:
  - Completely generic and unaware of cpufreq and cpuidle policies.
  - Determines when and *where* a task runs, i.e. on which cpu.

Existing Power Policies



- No coordination between power policies to avoid conflicting/suboptimal decisions.
  - Is it a problem?

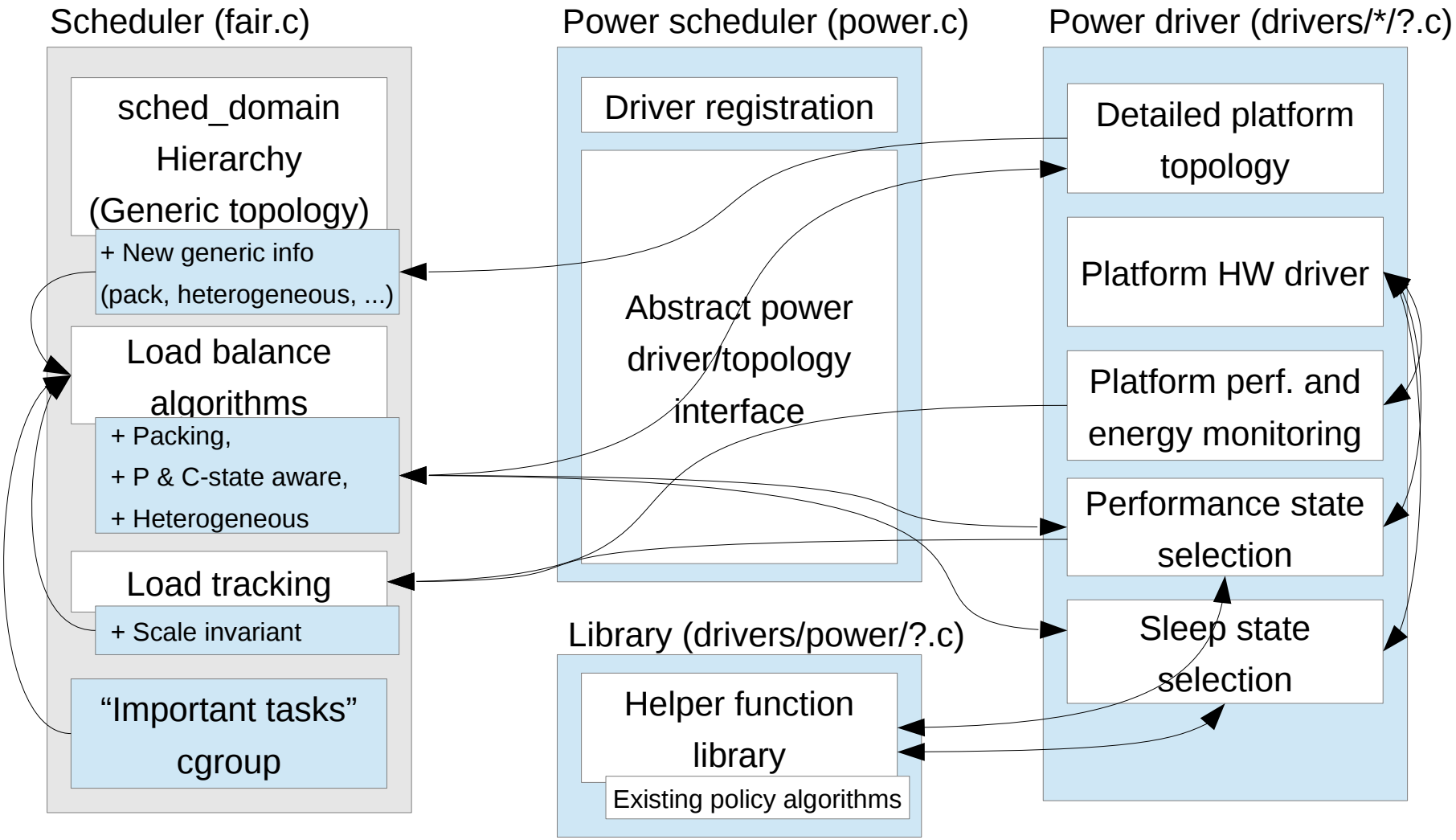
Issues

- Scheduler->cpufreq->scheduler cpu load feedback loop
  - From 3.11 the scheduler uses tracked load for load-balancing.
  - Tracked load is impacted by frequency scaling. Lower frequency leads to higher tracked load for the same task.
- Hindering new power-aware scheduling features
  - Task packing: Needs feedback from cpufreq to determine when cpus are full.
  - Topology aware task placement: Needs topology information inside the scheduler to determine the most optimal cpus to use when the system is partially loaded.
  - Heterogeneous systems (big.LITTLE): Needs topology information and accurate load tracking.

Wish-list

- Scale invariant load tracking
  - Fix scheduler->cpufreq->scheduler feedback loop
  - Better task packing
  - Needed for heterogeneous systems
- Topology awareness
  - Improve idle decisions
  - Scheduler frequency scaling awareness
  - Thermal/power budget management
  - Heterogeneous system (big.LITTLE) support

Power scheduler proposal



Power driver interface

- Platform agnostic scheduler interface:
  - The scheduler can only request information not HW state changes from power driver.
  - The scheduler provides hints to the power driver or hardware. Hints may be ignored.
  - Keeps platform specific topology/hardware information in the driver.
    - Detailed platform information is hard to represent in a generic (and useful) data structure in fair.c. It is even harder to design a one fits all policy.
  - Driver is supported by generic helper function library
    - Reuse common algorithms across drivers
    - Flexibility to have platform specific optimizations without bypassing existing frameworks (intel_pstate.c).

Proposed driver interface (scheduler)

API	Description
max_capacity(cpu)	Can the cpu go any faster? At highest available P-state.
increase_capacity(x)	Increase capacity by x hint. Go to higher P-state if possible. Driver may ignore x.
decrease_capacity(x)	Decrease capacity by x hint. Go to lower P-state if possible. Driver may ignore x.
task_boost(cpu)	Important task schedule boost hint. Power driver may give priority to this cpu in thermal or power constrained situations. For example for turbo mode.
get_best_wake_cpu()	Returns optimal wake-up target cpu when more cpus are needed.
get_best_sleep_cpu()	Returns the best cpu to idle when fewer are needed.
enter_idle()	Let the driver put the cpu to sleep.
load_scale(cpu)	Return tracked load scaling factor to compute scale invariant tracked load. Possibly P-state or PMU based.
init_sched_domain(cpu, level)	Returns sched_domain flags and variables for sched_domain initialization.

Proposed driver interface (driver)

API	Description
power_driver_register()	Register platform specific power driver.
idle_gov_menu()	“menu” idle governor heuristics from library.
idle_gov_ladder()	“ladder” idle governor heuristics from library.
freq_gov_ondemand()	“ondemand” freq governor heuristics from library.
freq_gov_pid()	intel_pstate.c style freq governor from library.
...	

V1 design feedback

- Don't use cpu_power to restrict scheduling.
  - Possible solution: Integrate packing directly into load-balancing logic.
- Some platforms have (partial) HW power management that may/will ignore OS requests.
  - Suggested solution: Abstract platform driver interface that gives hints rather than requests.
- We cannot have two captains (power vs. process scheduler)
  - Possible solution 1: Implement all policy details in fair.c adding a significant amount of complexity.
  - Possible solution 2: Abstract the policy decisions and move the decision to the power driver whenever possible. Provide helper function library to support power driver.

Summary

- Several problems to address and the solutions will affect each other.
- Patches to solve some of the problems individually have been posted on LKML, but never made any progress towards being accepted.
- A unified approach is needed.