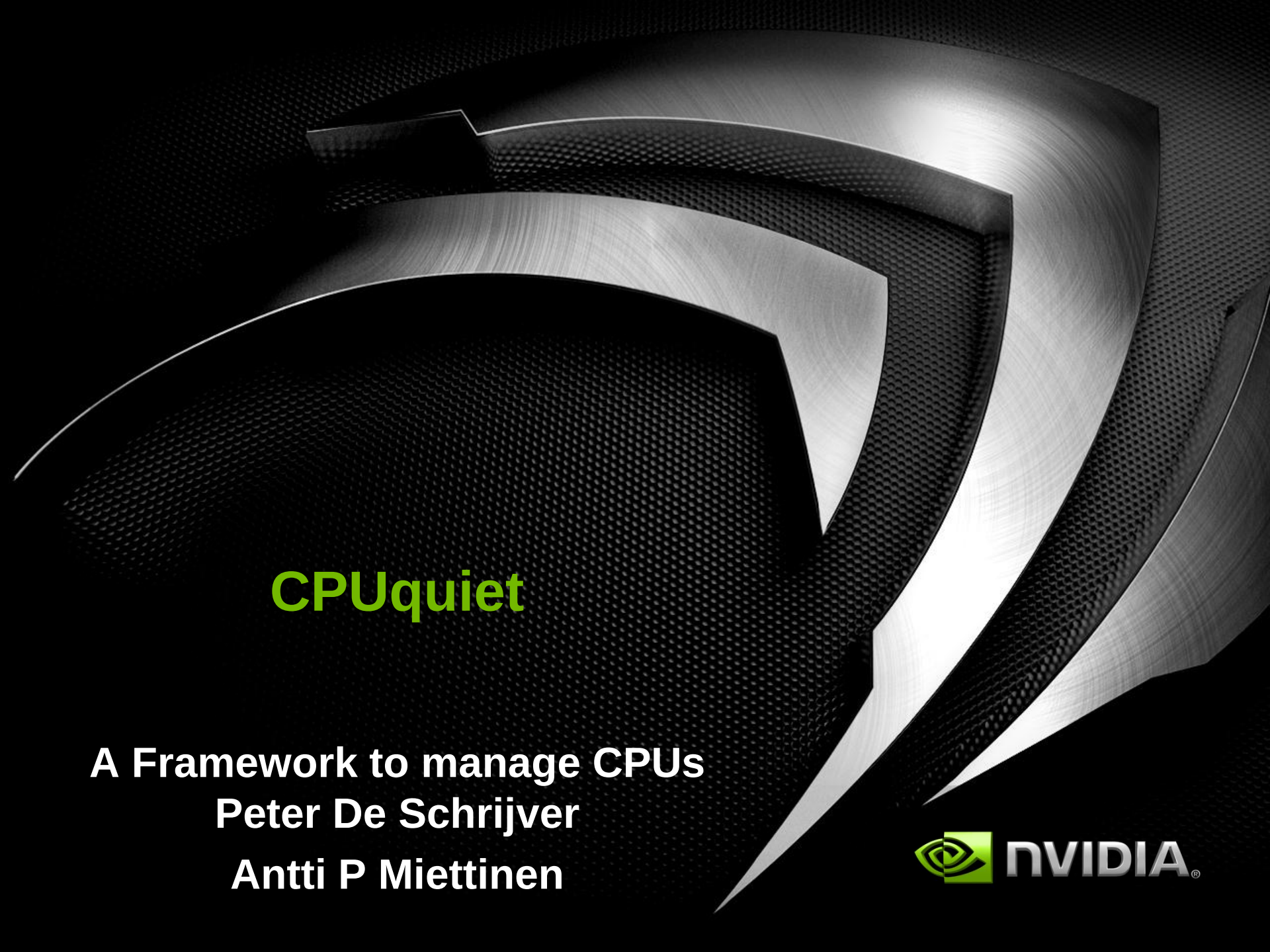

CPUquiet
A Framework to manage CPUs
Peter De Schrijver
Antti P Miettinen
NVIDIA

History (1)

- Tegra30: 2 CPU clusters
  - G cluster: 4 high performance Cortex A9 cores
  - LP cluster: 1 low leakage Cortex A9 core
  - Only 1 cluster can be active at a time
- To enable use of LP cluster, we must allow only 1 CPU to be active
  - No interrupts or other wakeups possible for other cores
- Solution: CPU management

History (2)

- Other reasons for CPU management
  - G cluster in single core mode can run at a higher frequency than in multi core mode
  - Peak current constraints
    - Max current per core is temperature dependent
    - More cores online means higher worst case current
    - Regulator might not be able to deliver enough current to supply all cores at max frequency at the max temperature
    - Depending on the workload it might be better to offline cores rather then lower the max CPU frequency in hot conditions
  - Power savings:
    - Wakeups limit the efficiency of cpuidle states
    - Intentionally cap performance

CPU management implementation

- Use hotplug to online/offline cores
- Implemented as ’autohotplug’ mechanism in arch/arm/mach-tegra/cpu-tegra3.c
  - Implements policy and mechanism in 1 file
  - Uses the per cpu frequency targets set by cpufreq as a measure of load
  - Recently also nr_running() has been added as an input to the policy
    - Some changes not upstream
  - Handles both onlining/offlining cores and cluster switch

Problems

- Experimenting with policies is difficult
- Hotplug is about 1000 times slower than power gating (turning off the core)
  - Even though came from about 150ms to about 10ms!
    - Can be a lot faster
  - Causes power inefficiencies because we can't track load changes accurately
  - Causes performance loss, due to slow reaction on load increase
  - Complicates policy design
- Doesn't scale to other similar systems (eg. ARM big.LITTLE switching)

CPUquiet goals

- Decide which CPUs should be available
- Separate policy and mechanism
- Allow for runtime selection of the policy (governor)
- Provide the same features and performance as our current ‘autohotplug’
- Allow for other mechanisms than hotplug to quiesce CPUs
- Handle cluster switching outside this framework

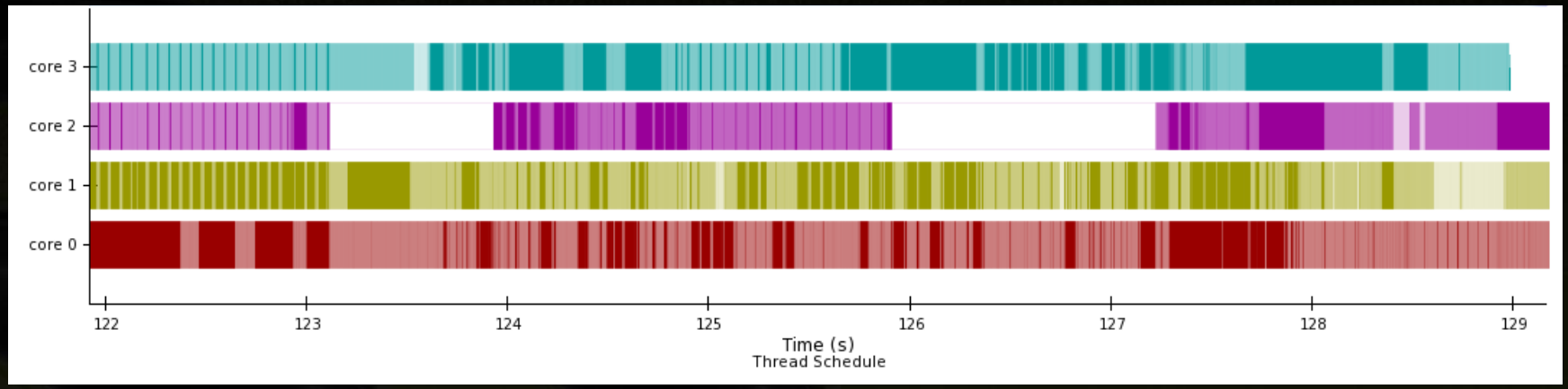
CPUquiet implementation

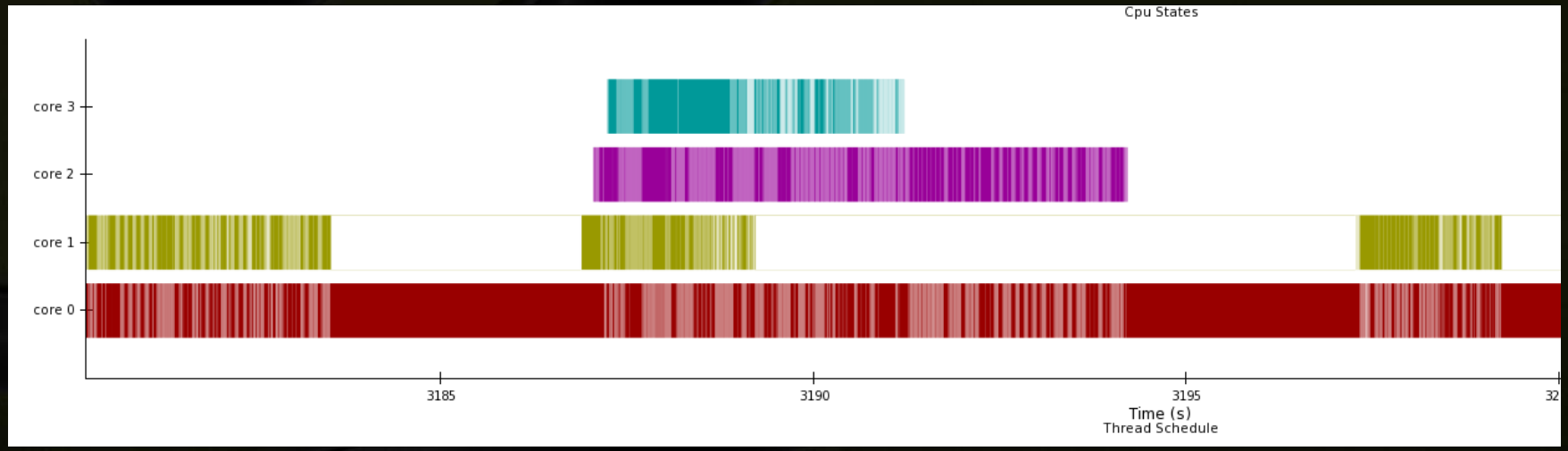
- Framework with the following components
  - Cpuquiet driver
    - quiesce_cpu(): ensures the CPU will not be used by the kernel
    - wakeup_cpu(): makes the CPU available to the kernel again
  - CPUquiet governor: implements policy
  - Available drivers:
    - Tegra: uses hotplug and handles cluster switching

CPUquiet implementation (2)

- Available governors:
  - Userspace
  - Balanced: uses CPU load and nr_running() to implement the same policy as ’autohotplug’
  - Runnables: use runnable threads as input

Policy does have an effect





CPUquiet future work

- Make part of mainline linux
  - For now look at https://github.com/pboonstoppel/
- Drivers for other platforms (e.g. ARM big.LITTLE)
- Improve hotplug performance or design a new mechanism to quiesce and wakeup CPUs
- Integration with PM constraints
- Move cluster switching into the framework?