Scheduler Options in big.LITTLE Android Platforms

Mike Anderson
mike@theptrgroup.com
http://www.theptrgroup.com

What We'll Talk About

- big.LITTLE ARM architectures
- Cluster Scheduling
- In-Kernel Switcher (IKS)
- Global Task Scheduler
- Selecting the right CPU for the job
- Energy-aware scheduling
- Anything Android specific?
- Summary

Traditional ARM Architectures

- Traditional ARM multi-core SoCs are symmetric
  - All cores are the same variety such as A9s or A15s
- This makes scheduling very straightforward
  - Run the task on any CPU that's available
- CPU migration is probably not a problem for most Android users
  - Hitting a cold cache is not really much of a problem on a handheld device because of the human in the loop

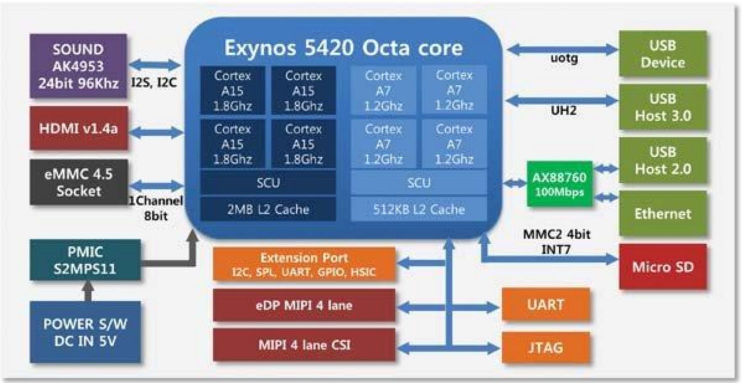
Power Management Issues

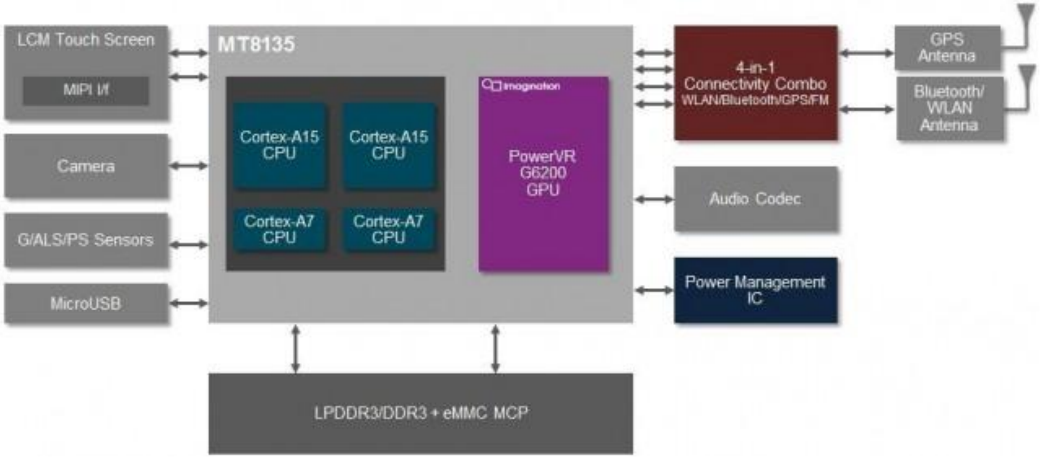
- Assuming that the CPU is already capable of power gating, DVFS is a significant player in power management
- Given a fully symmetric SoC, opportunities for power scaling were largely limited to DVFS in the kernel
  - Dissipated power over time = $C*V^2*A*f$
    - C is capacitance of gates
    - V is voltage (note this is a *squared* term)
    - A is activity factor (average number of switching events in the transistors)
    - f is the frequency
- Changing frequency requires a change in voltage
  - But, all devices on the bus need to be aware and capable of the new frequencies
- However, even with DVFS, the Cortex A9 would still use a lot of power even if idle

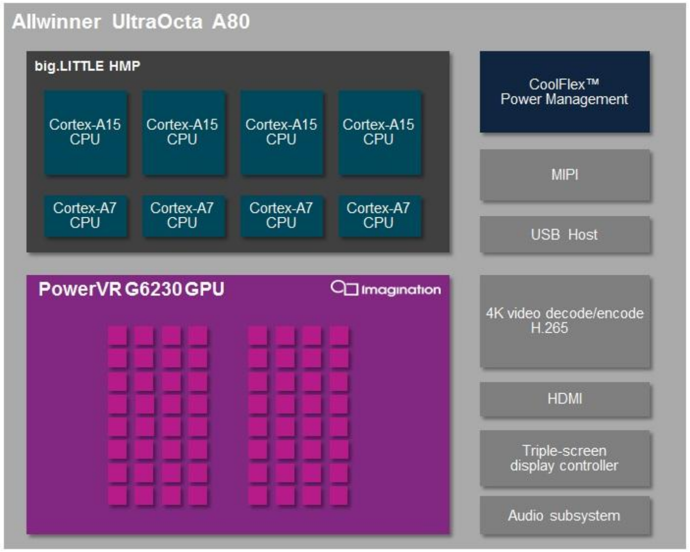
big.LITTLE ARM Architectures

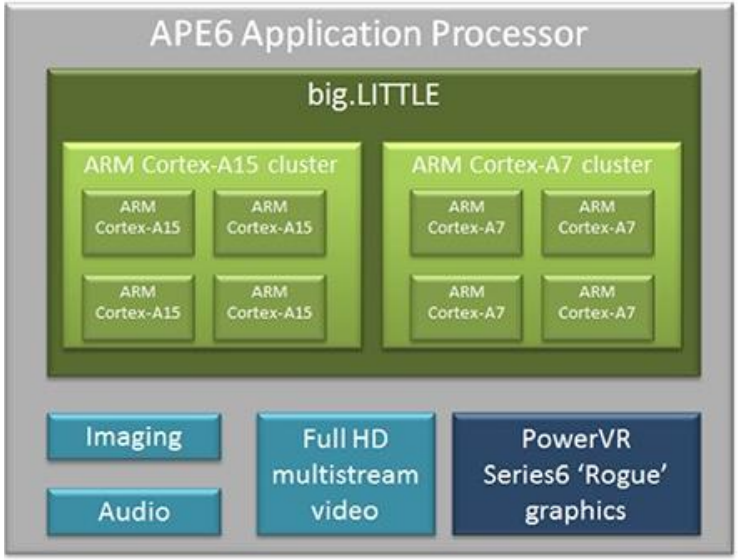
- In 2011, ARM introduced the Cortex A7
  - 1.5x the processing capability of the A8 with 1/5 the power consumption
  - The intent was to pair the A7 with the A15 as a way of saving power
- As the workload dropped below a certain point, the A15 cores would be shut off and the A7 cores would take over
- The first commercial implementation of big.LITTLE was the Samsung Exynos 5410 used in the Samsung Galaxy S4

Example big.LITTLE Implementations









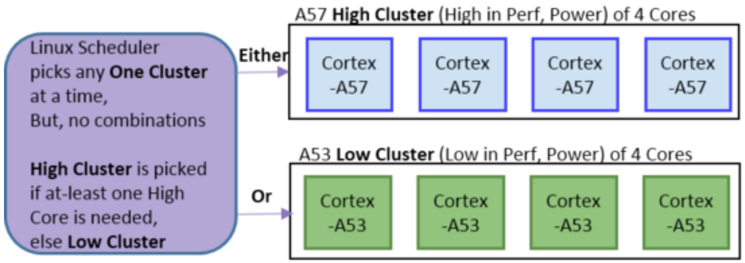
Source: The respective manufacturerc

Cluster Switch Scheduling

- In the initial implementation of the big.LITTLE architecture, the CPU was cluster scheduled
  - Either the big or the LITTLE cluster was active, but not both
- Significant impact on performance during the cluster switch
  - Cache coherency hardware is a must
  - You still have inefficiencies if all of the processors in the cluster were activated, but the workload didn’t need all of them at once

Cluster Switching and Android

- The cluster switch approach was the default scheduler in Android 4.2.2



Source: linaro.org

- This was commonly encountered on Samsung tablets and international phones using the Exynos CPU family
  - International versions of Samsung phones
- Android uses cgroup and DVFS mechanisms to help make cluster decisions
- CPU affinity can be used to override scheduling decisions

Problems with Cluster Switching

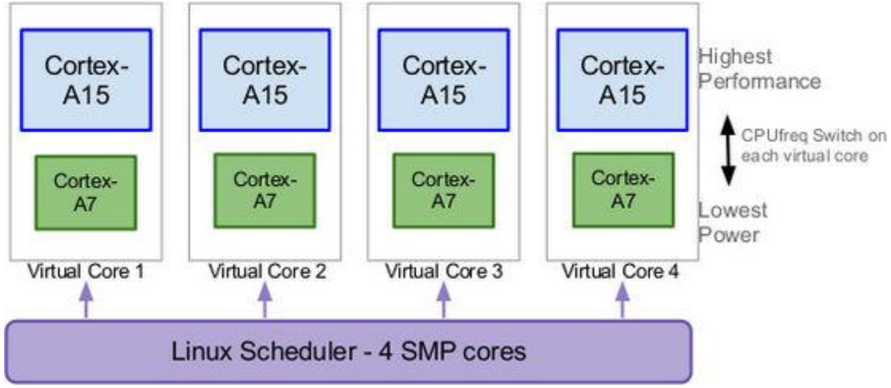
- Cluster switching does conserve power, but not as much as people hoped
- Problems with having to do an all-or-nothing approach to switching the cluster left developers thinking there must be a better way
- Hardware implementation problems in early big.LITTLE SoCs precluded all other alternatives
  - But, that doesn't mean kernel developers give up ☺

ARM/Linaro to the Rescue

- Working in conjunction with ARM and using a new ARM reference board, Linaro developers came up with two alternative approaches for using big.LITTLE processors
- In the first, the big and LITTLE cores are teamed up into virtual CPUs
  - One big and one LITTLE core per VCPU
- Whether the application runs on the big or the LITTLE core is based on CPU load for that task
  - However, we still have at most four cores running at any point in time on an octa-core platform
  - Referred to as In-Kernel Switching (IKS) or CPU Migration

Example IKS

- This approach groups processors into collections of virtual cores where big and LITTLE processors are teamed together

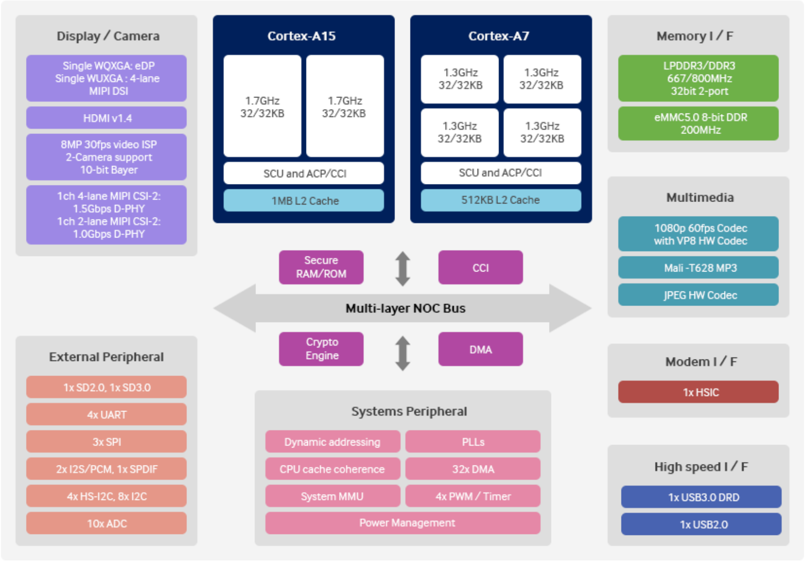


Source: linaro.org

- Again, the CPU load of the task is used to determine if the big or LITTLE core actually runs the task
- This approach allows a mixture of big and LITTLE cores to run as needed
  - However, we're still only executing at most half of the cores at any point in time

What about “Unbalanced” Processors?

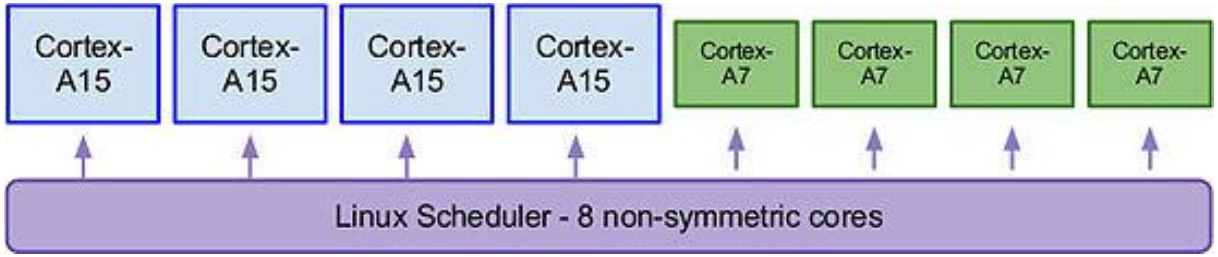
- With processors like Samsung’s hexa-core, multiple LITTLEs can be teamed with a single big
- While this works, it doesn’t provide quite the granularity that we’d like to have in scheduling



Source: samsung.com

- IKS patch came out for the 3.10 Linaro Stable Kernel (LSK) and 3.14 Linux mainline
- Google does not ship a kernel capable of IKS at this time

Getting Maximum Performance



Source: linaro.org

- Given that you've got both big and LITTLE cores on the platform, it might be nice to be able to use all of them at the same time for a burst of computation
  - That is the goal of the Global Task Scheduler a.k.a. big.LITTLE MP
- In this option, each processor core is active and can be scheduled
  - The previous load for the task determines which core the task runs on
  - Again, CPU affinity can be used to lock the task to a particular core

Problems with big.LITTLE MP

- The original big.LITTLE MP was largely developed and put forward by ARM
  - Unfortunately, there were a lot of places that needed to be touched in the kernel to make it happen
  - Nonetheless, the GTS scheduler did get deployed in several of the Samsung devices like the international Galaxy S4/S5 and some Chromebooks
- Since many of the changes for this approach would affect non-ARM architectures as well, this approach was rejected and a new approach is being developed

Energy Aware Scheduling (EAS)

- EAS is a set of kernel extensions that introduces an energy-based model for power-performance control and task scheduling
  - The scheduler will be the focal point for power-performance decisions rather than cpufreq or cpuidle subsystems
- The goal is use a scheduler-driven policy and a small set of well-defined tunables to simplify power/performance management

EAS #2

- Using cpufreq makes energy policy creation rather complicated
  - cpufreq, cpuidle and the scheduler tend to get in each others ways
- EAS aims to provide tools that assist with the creation and qualification of an energy model
  - This includes the quantification of energy usage per workload as well as power-performance tuning
- EAS is the culmination of a series of discussions on the LKML as well as discussions at various conferences

EAS Moving into the Kernel

- EAS is not a fait acompli at this point
- There are several board tracks that need to be addressed before the work is complete including:
  - SCHED-CORE
    - Introduces the CPU energy model
    - Applies the energy model for load-balance decisions
    - Applies the energy model for power-performance control
    - Modifications to the CFS to accommodate the energy model approach

Kernel Tracks #2

- SCHED-CPUFREQ
  - Modifications to the cpufreq code to enable the scheduler to control DVFS OPP transitions
  - Creation of a simple, scheduler-driven policy for DVFS
    - Creation of a set of tunables to provide the knobs needed to enable power/performance options
- SCHED-CPUIDLE
  - Modifications to make the scheduler aware of all of the idle states supported by CPUs in the system
    - Includes the cost implications of entering and exiting power states as well as idle state tracking
  - Remove any redundant idle state-specific data

User-Space Tools

- Idlestat
  - The purpose of idlestat is to measure how long we've been in the various idle and operating states
- Uses FTRACE function to monitor entry and exit of C- and P-state transitions over time
  - Tracks the times for entry and exit of the states as well as any raised IRQs
- Following a successful run, the trace data is parsed to show:
  - The total, average, min and max of time spent in each C- and P- state
  - Tracks the same stats for when all of the CPUs in a cluster where in the same C-state per cluster
  - Tracks the number of times an IRQ cause a CPU to exit the idle state on a per-CPU and per-IRQ basis
- While there is some overlap with powertop, idlestat is designed to be non-interactive and provide more details on state entry and exit
- Source code git tree is available from:
  - http://git.linaro.org/power/idlestat.git

User-Space Tools #2

- Workload generator
  - Based on rt-app, the workload generator emulates typical mobile device use cases and gives runtime information
  - Uses JSON files for describing the use cases
- Allows you to create a simulated work load to capture the information from idlestat for creating a power-performance policy
- Linaro extensions to rt-app are available here:
  - https://git.linaro.org/power/rt-app.git

Current EAS status…

- The EAS kernel is available as a back-port to the currently available Linaro Stable Kernel v3.10
  - https://git.linaro.org/kernel/eas-backports.git
- Relevant discussions for EAS development can be found on the eas-dev public mailing list at http://lists.linaro.org/mailman/listinfo/eas-dev
- There will be publically open, bi-weekly telephone calls for purposes of discussing progress
- For more information, go to https://wiki.linaro.org/WorkingGroups/PowerManagement/Resources/EAS

  and

  https://rt.wiki.kernel.org/index.php/Energy_Aware_Scheduling

Anything Android Specific?

- Fortunately, no
- Since the Android kernel is really the Linux kernel, these changes apply equally across all of the operating systems using the Linux kernel
- When can we expect to see these changes?
  - The IKS was mainlined in 3.14
  - GTS is dead except for the few manufacturers that deployed it
  - EAS is still a work in progress

Summary

- Power management is an ever-important topic for handheld devices
- The scope of big.LITTLE processors will likely continue to expand as more silicon vendors embrace the technology
- Current Android uses the cluster-based approach to scheduling
- The IKS is available in mainline a/o 3.14
- EAS promises to be the best overall solution
  - Only time will tell