

Implications of Per CPU switching in a big.LITTLE system

Achin Gupta

ARM Ltd.

achin.gupta@arm.com

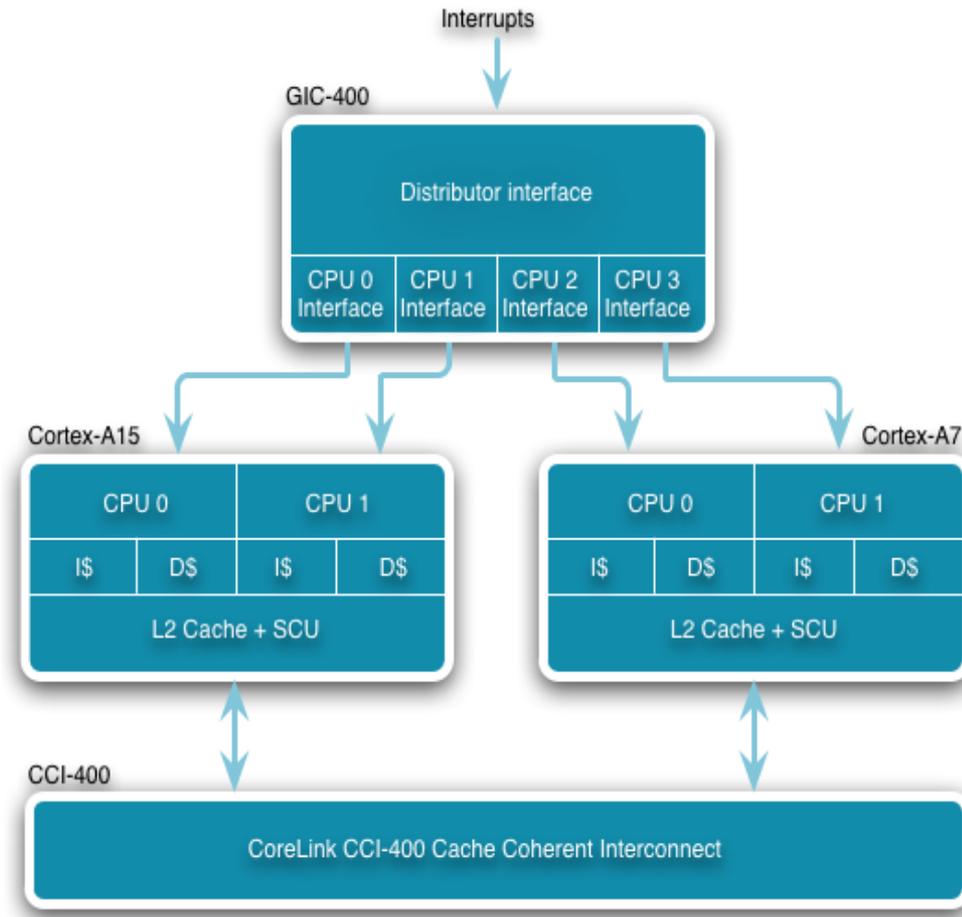


Agenda

- ARM big.LITTLE subsystem overview
- big.LITTLE Execution modes overview
- The Reference Switcher
 - Interoperability with Linux OSPM
- The Integrated Switcher
 - Interoperability with Linux OSPM
- Migration models and micro-architectural differences
 - Cache topology differences
 - PMU implementation differences
 - Shared Interrupt controller implications
 - Miscellaneous implementation defined differences
- Optimizations

ARM big.LITTLE subsystem overview

- Programmer's view
 - High performance Cortex-A15 cluster
 - Energy efficient Cortex-A7 cluster
 - Fully cache coherent via CCI-400
 - Signal pathway via shared GIC-400



big.LITTLE execution modes overview

- Two broad types
 - Migration
 - MP
- Migration has levels of granularity
 - Cluster migration
 - CPU migration
- Relative complexity scales from Migration towards MP
 - Migration modes provide benefit at reduced system software complexity for symmetric topologies
 - Asymmetric topologies are also supported...ish
 - MP mode provides flexibility and optimal utilization with higher system software complexity for all topologies

Cluster migration

- Only one cluster is ever on
 - Except briefly during a cluster switch
 - End-to-end raw switching interval is ~30 Kcycles
- Cluster selection driven by OS power management
 - DVFS algorithms mitigate load by selecting a suitable operating point
 - A switch from the Cortex-A7 cluster to Cortex-A15 cluster is an extension of the DVFS strategy
- Load monitoring is done at the cluster level
 - Linux cpufreq samples load for all CPUs in the cluster
 - Selects an operating point for the cluster
 - Switches clusters at terminal points of the current cluster's DVFS curve

CPU migration

- Paired little and big CPU operation
 - Each little CPU can switch to its big counterpart
 - Each CPU switches independently of other CPUs
- CPU selection is driven by OS power management
 - DVFS algorithm monitors per-CPU load
 - Operating point selection is done independently per-CPU
 - When a little CPU cannot service the incumbent load a switch to its big counterpart is performed
 - The little processor is switched off at this point
 - The big processors are used opportunistically in this manner

The Reference Switcher

■ Description

- Uses ARM Virtualization Extensions to run an SMP OS on a big.LITTLE system
- Adopted the simplest approach to use a big.LITTLE system in an OS agnostic way
- Switches payload software execution synch/asynchronously between clusters in an optimized way
- Masks micro-architectural differences between Cortex-A15 and Cortex-A7
- Integrated with Linux DVFS subsystem to demonstrate
 - Load driven cluster migration
 - Switching hysteresis control algorithms
- Still the fastest way to use a big.LITTLE system.....or find if the system is usable

Reference Switcher

- Tradeoffs
 - Good at implementing mechanics not policy
 - Would need to rely on heuristics in absence of Linux support
 - Hard to perform a cluster switch in HYP mode while Linux might be idling/hotplugging cpus
 - No way to virtualise Idle & OPP tables. Expected to be provided by the Power controller each time they need to be used
 - Works best with a symmetric big.LITTLE topology (same number of cores)
 - Asymmetric topologies can be used but require scheduler involvement
 - Relies on payload software to 'auto detect' features to hide micro-architectural differences
 - Needs intelligence to efficiently use CCI

Reference Switcher

- Interoperability with Linux OSPM
 - Needs Linux to implement the policy and initiate a cluster migration
 - DVFS framework needs cluster awareness
 - Idle framework needs the same
 - Notifiers are needed to switch between OPP and Idle tables
 - Interoperability with idle & hotplug is a big policy dilemma
 - Unpredictable behaviour if in-flight cache maintenance operations are migrated
 - Can switch only when other cpus are in a known stable state
 - Migration policy difficult to implement if clock and voltage domains are per-cpu

Reference Switcher

- Hotplug policy
 - Offline cpus should be left as they are
 - In-flight hotplug operations should be allowed to complete
 - Further hotplug operations should be disabled till the cluster migration does not complete
 - Notifiers should be used to implement this

contd....

Reference Switcher

■ Idle Policy

1. Preserve idle state of as many cpus as possible
 - Needs help from the Power controller to keep idle cpus idle
 - Send an IPI to running cpus to either:
 - Begin a cluster migration
 - Enter a “quiet” state and let the reference switcher take over
 - Inflight idle operations would be redundant
 - Complicated protocol to preserve idleness
 - Power controller prevents idle cpu wakeup temporarily

Reference Switcher

- Idle Policy

- 2. Wakeup all idle cpus

- Needs Linux to wakeup all idle cpus and make them either initiate a cluster switch or enter the “quiet” state
 - Simpler & Fast to implement. Could have a detrimental effect on power savings as cpus in possibly deeper c-states will need to be woken up

Reference Switcher

- To sum up.....
 - Reference Switcher helps in providing the mechanics of performing a cluster switch & maintaining the illusion of an SMP system
 - Needs help from the OS to get even the mechanics of power management working
 - Has lower impact on policies but greater impact on mechanics.
 - Works well if the DVFS plane spans the cluster. Things get interesting when the clock and voltage domains are per-cpu
 - Policy implications on power and performance costs not benchmarked
 - Needs further optimizations to keep inbound caches warm by utilizing the CCI. This is critical to make cluster migration practical

Integrated Switcher

- Why migrate a cluster when a cpu will do!
 - Do load calculation on a per-cpu basis
 - Migrate only the cpu as per its needs (power/performance)
 - No need to worry about other cpus for mechanical bits
- cpuidle and cpufreq drivers still need to be cluster aware
- Inbound caches are kept warm implicitly
- Simple and efficient implementation. A cpu switch can be thought of as a combination of cpu hotplug and idle
 - Outbound cpu needs to be hotplugged
 - Inbound cpu needs to be resumed from a suspend
- Critical to keep track of cpus
 - Common layer is needed for tracking cpu migration, hotplug & idle
 - Export this common view to Linux OSPM framework & GIC driver

Integrated Switcher

- Re-uses Linux code to save/restore architectural state and interrupt controller context
- Warm reset path needs to be cluster aware
- Lots of policy options.
 - A cpu might be shutdown or hotplugged on one cluster but it might make sense to wake it up on the other
 - Amongst other metrics that cpuidle needs to worry about before deciding the target c-state, it will have to look out for cpu migration as well
 - A cpu switches in response to a per-cpu load calculation which is perfect when it is in its own voltage plane driven by a per-core clock
 - Things get interesting when the clock and voltage domain spans the cluster
 - A hotplug operation could result in a cluster shutdown

Micro-architectural differences

- Cache topology differences (D-Side)

Processor	L1 D-cache	L2 Unified-Cache
Cortex-A15	2-way set associative of 32KB	16-way set associative of 512KB, 1024KB, 2048KB or 4096KB
Cortex-A7	4-way set associative of 8KB, 16KB, 32KB or 64KB	8-way set associative of 128KB, 256KB, 512KB or 1024KB

- The Reference Switcher maps one cache topology to another
- The Integrated switcher will rely on auto-detecting this information or maintaining per cluster cache topologies if required
- Affects only set/way operations. They are expected to be called only during power down e.g. idle or hotplug

Micro-architectural differences

- Cache topology differences (I-Side)

Processor	L1 I-cache
Cortex-A15	64 byte cache lines in a PIPT cache
Cortex-A7	32 byte cache lines in an aliasing VIPT cache.

- The Reference Switcher exports the Cortex-A7 view to Cortex-A15 and helps bridge the gap at the cost of redundancy
- Integrated Switcher is not responsible for this bit. Linux needs to ensure maybe through the use of device trees that this difference is catered for

Micro-architectural differences

■ PMU differences

- Cortex-A15 implements 6 event counters compared to 4 implemented by Cortex-A7
- There are numerous differences in implementation defined events
- Reference switcher uses Virtualization extensions to allow use of the PMU across both clusters
 - Perf sees and uses events that it detects on the boot cluster
 - Events are saved and restored across a cluster switch. Perf is responsible for using only the events which are common across the two clusters
 - HVC api is available to use both PMUs. Each PMU's state is preserved across a cluster switch

Micro-architectural differences

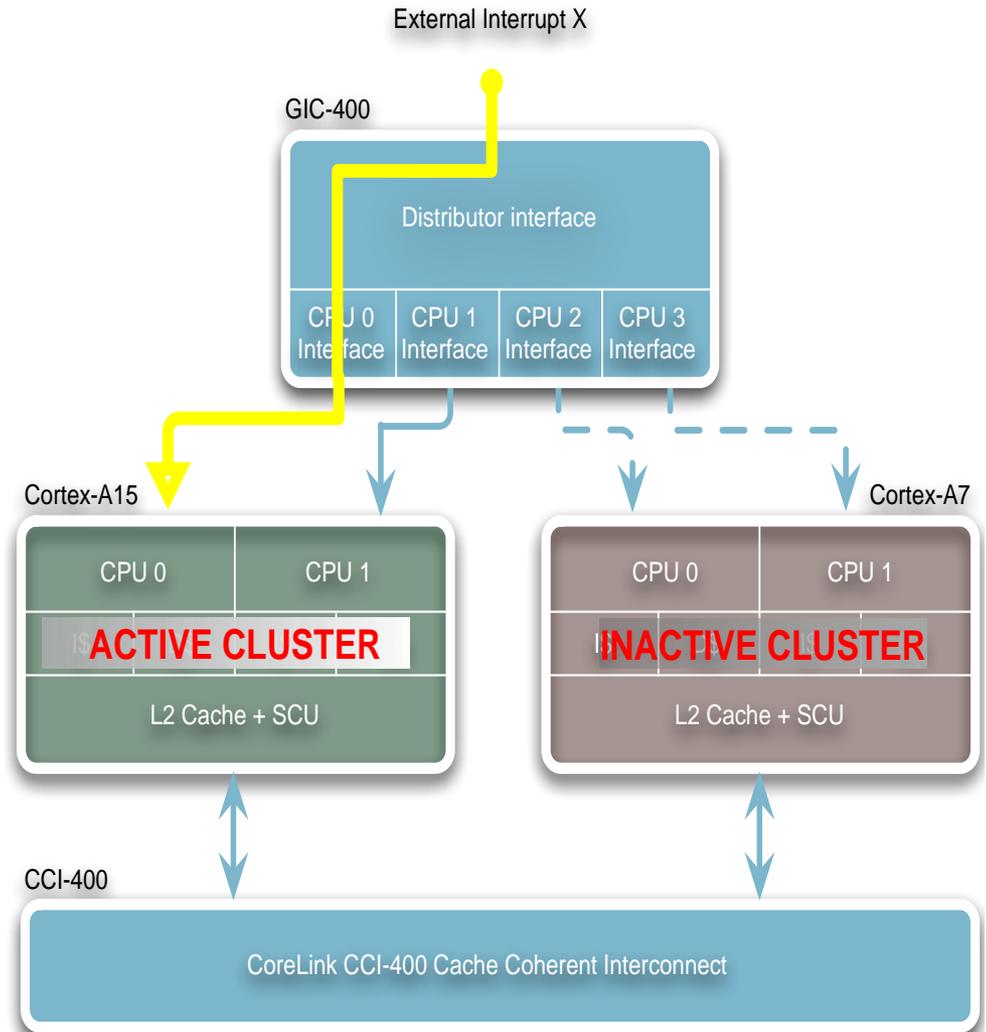
- PMU differences contd..
 - Integrated switcher needs the perf backend to maintain separate event "buckets" for each cluster
 - Count events available on Cortex-A15 only on it and ditto for Cortex-A7
 - Separate counts for common events e.g. cycle counter
 - Scales well for big.LITTLE MP as well

Micro-architectural differences

- Implementation defined register differences
 - Common registers with different bit implementations e.g ACTLR, L2CTLR
 - Registers present on one and not on the other
 - L2ACTLR, L2PMRR etc in Cortex-A15
 - Copy TLB RAM to registers (CDBGTD) in Cortex-A7
 - These registers are not used by Linux but need to be preserved if required across a migration

Micro-architectural differences

- Shared vGIC
 - CPU interfaces are numbered linearly
 - CPU ids are repeated
 - Mapping between the two needs to be maintained
 - Reference Switcher virtualizes accesses to the GIC Distributor
 - Integrated Switcher maintains internal map.



Optimizations

- **Pipelining**
 - Bring the inbound core out of reset beforehand
 - Secure firmware setup completes while context is saved
 - Device, Coherency and MMU setup is costly
 - Power down outbound cpu while inbound restores context
 - Allows the inbound to tap into the warm outbound caches
 - Especially important when the last cpu in a cluster migrates causing an outbound cluster shutdown
- **Turn on MMU & I caches asap after a reset**
 - CCI snoops can be turned on later
- **Use DSBs only when required**
 - A Data Synchronization barrier spans the inner shareability domain
 - Waits for outstanding transactions to complete and can cause stalls

Optimizations

- Enable XN bit for pages which do not contain instructions
Pre-fetching can be counter productive
- Avoid accesses to literal pools. Replaced with MOV/MOVT for loading constants
- Use of virtual GIC interfaces can be avoided if Linux can trigger a cluster switch
- Target Interrupts to both clusters to avoid GIC s&r overhead
- Burst accesses to Device memory are better than word accesses

Thank You!

Questions.....??