In Kernel Switcher: A solution to support ARM's new big.LITTLE technology

Presenter: Mathieu Poirier

web: www.linaro.org
email: mathieu.poirier@linaro.org

What is big.LITTLE?

- A system that contains two sets of architecturally identical CPUs.
- CPUs differ in the power and performance they yield.
- Similar architecture allows to:
  - Run the same software transparently on all CPUs.
  - Migrate from one CPU to another transparently.

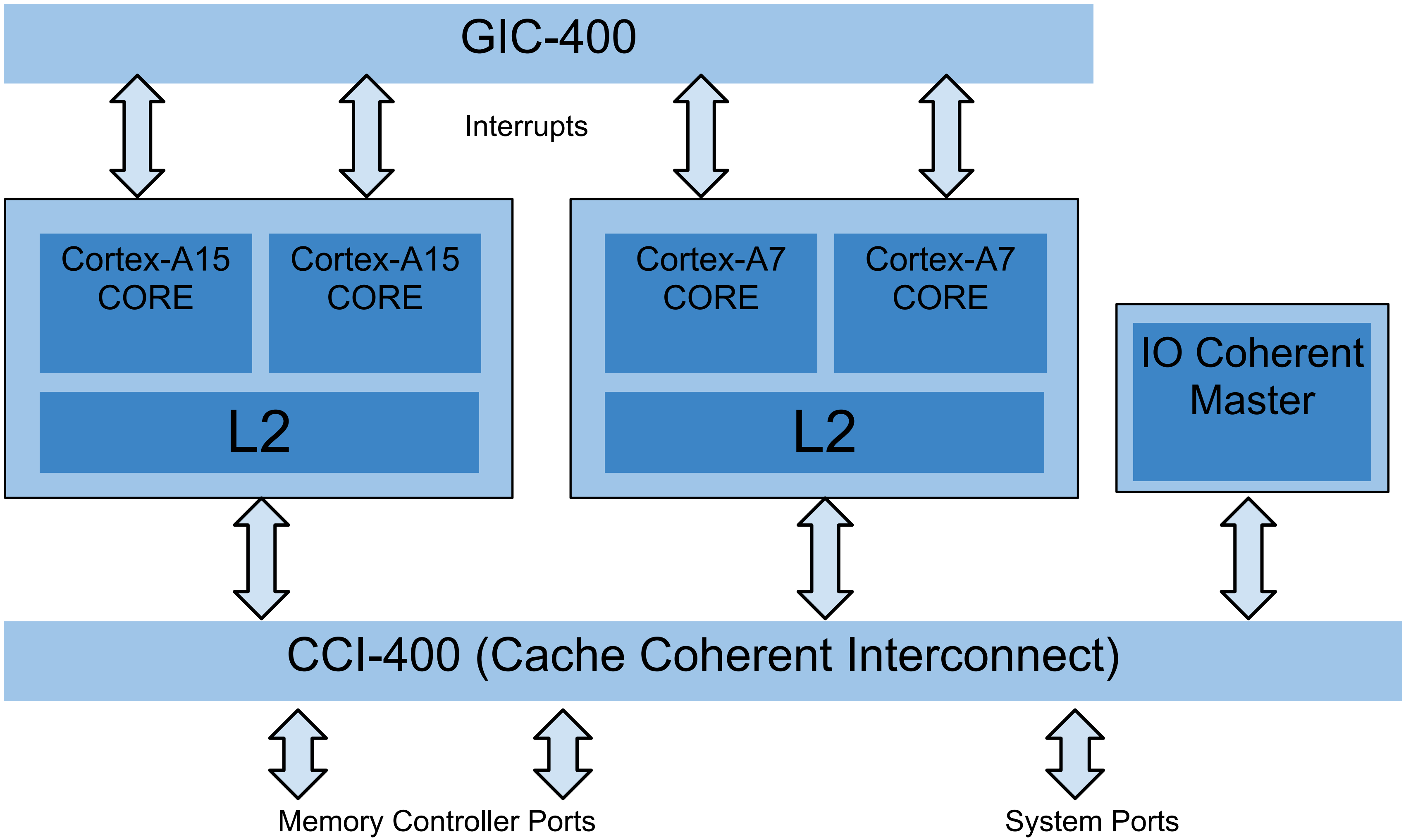
TC2 - ARM's big.LITTLE implementation

- Has a cluster of Cortex-A15 processors (big) and a cluster of Cortex-A7 processors (LITTLE) in the same system.
- Cortex-A7 and A15 are architecturally similar - ARM v7A.

- Processor caches are kept coherent using a cache coherent interconnect (CCI-400 on TC2).

- A shared Generic Interrupt Controller(GIC-400 on TC2) is used to migrate interrupts between any cores in the big or LITTLE clusters.

What does big.LITTLE look like?



* Picture by ARM LTD.

What is the idea behind big.LITTLE?

- The goal is to use the A15 cluster for CPU intensive task and the A7 cluster for low power task, for example:
    - Gaming - A15
    - Web page rendering: A-15
    - Texting - A7
    - Email - A7

- Provide a balance between performance and power efficiency.

What is being done at Linaro

- We currently have 2 big.LITTLE projects:
  - Heterogenous Multi Processing (HMP).
  - In Kernel Switcher (IKS).

- We can switch between them on the fly !
  - IKS can be enabled in the kernel config.
  - Or on the kernel command line.
  - Or at run time from sysfs.

Heterogeneous Multi Processing (HMP)

- All cores in the system can be used at the same time.
- Scheduler needs to be aware of different CPU processing power when scheduling.
- Higher peak performance for some workloads but harder scheduling problem for the kernel.
- Currently being developed in collaboration with the community.

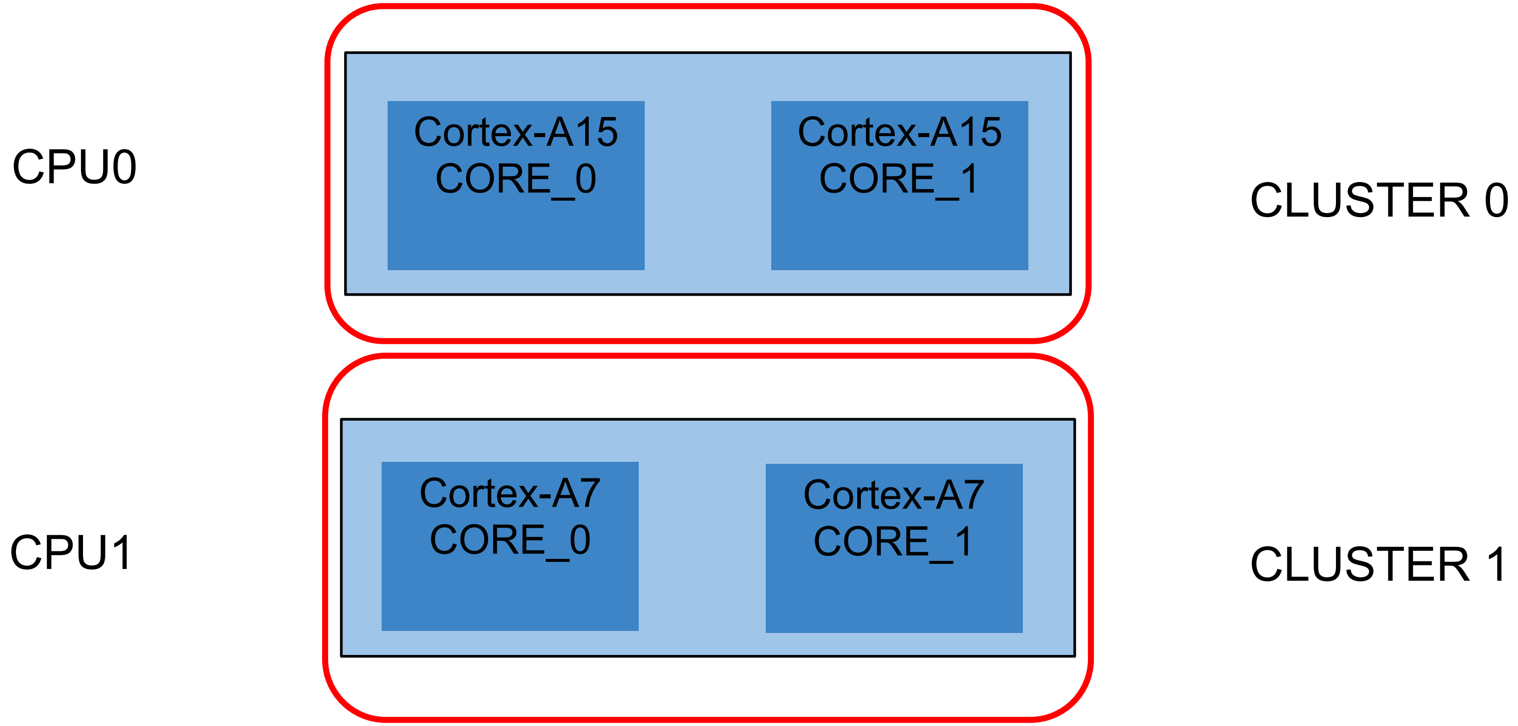
www.linaro.org email: contactus@linaro.org

In Kernel Switching(IKS) at Linaro

- A7 and A15 CPU from each cluster are coupled together to form a "virtual" CPU.
- All virtual CPUs have the same processing capabilities.
- The kernel core doesn't need to know about the asymmetric nature of the b.L architecture.

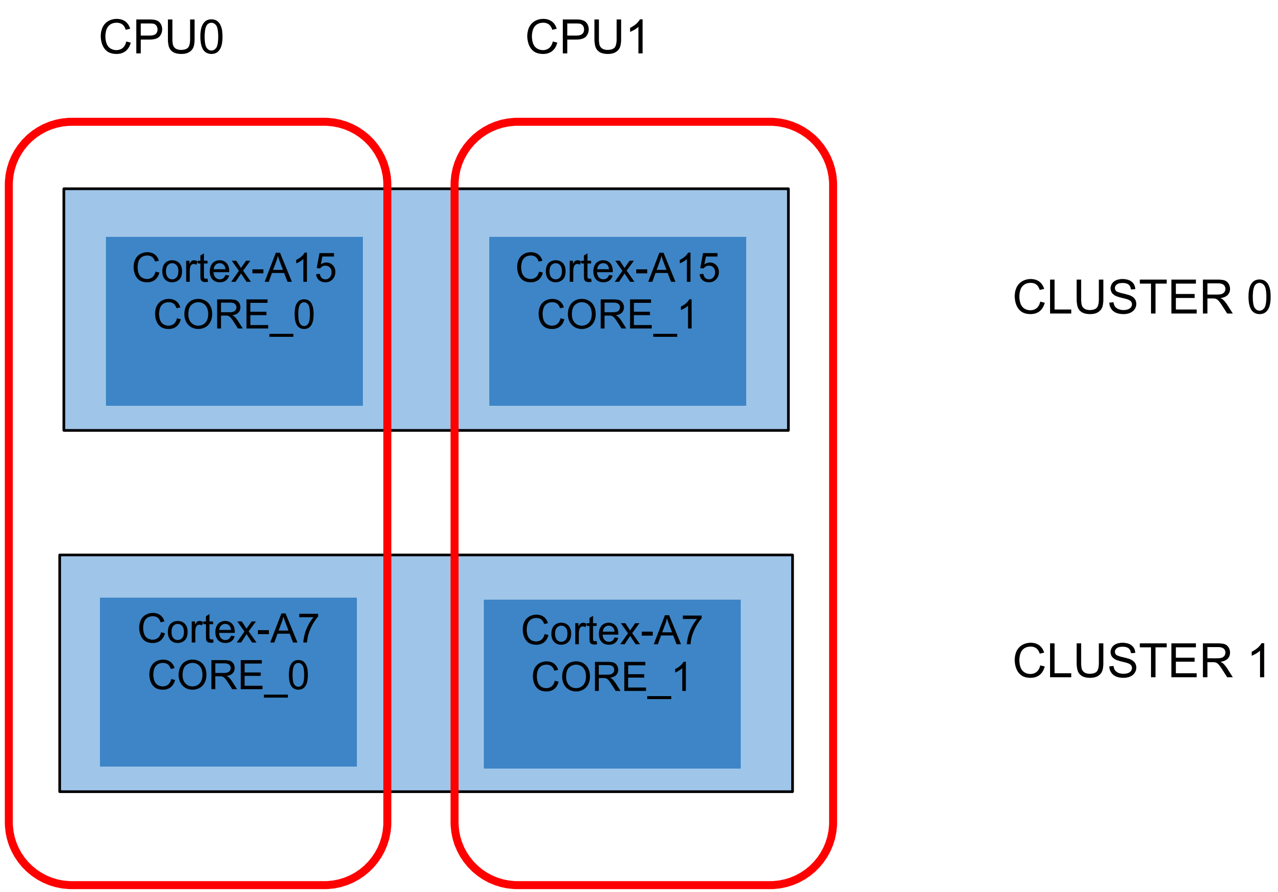
- Only one core is active in a given virtual CPU.
- Decision to move from one core to another is taken at the CPUfreq driver level.
- Released to Linaro partners in December of 2012.

One possible solution



- Inefficient - granularity is too coarse.
- Synchronisation period needed before switching.

What Linaro has implemented



IKS - Creation of the virtual CPUs

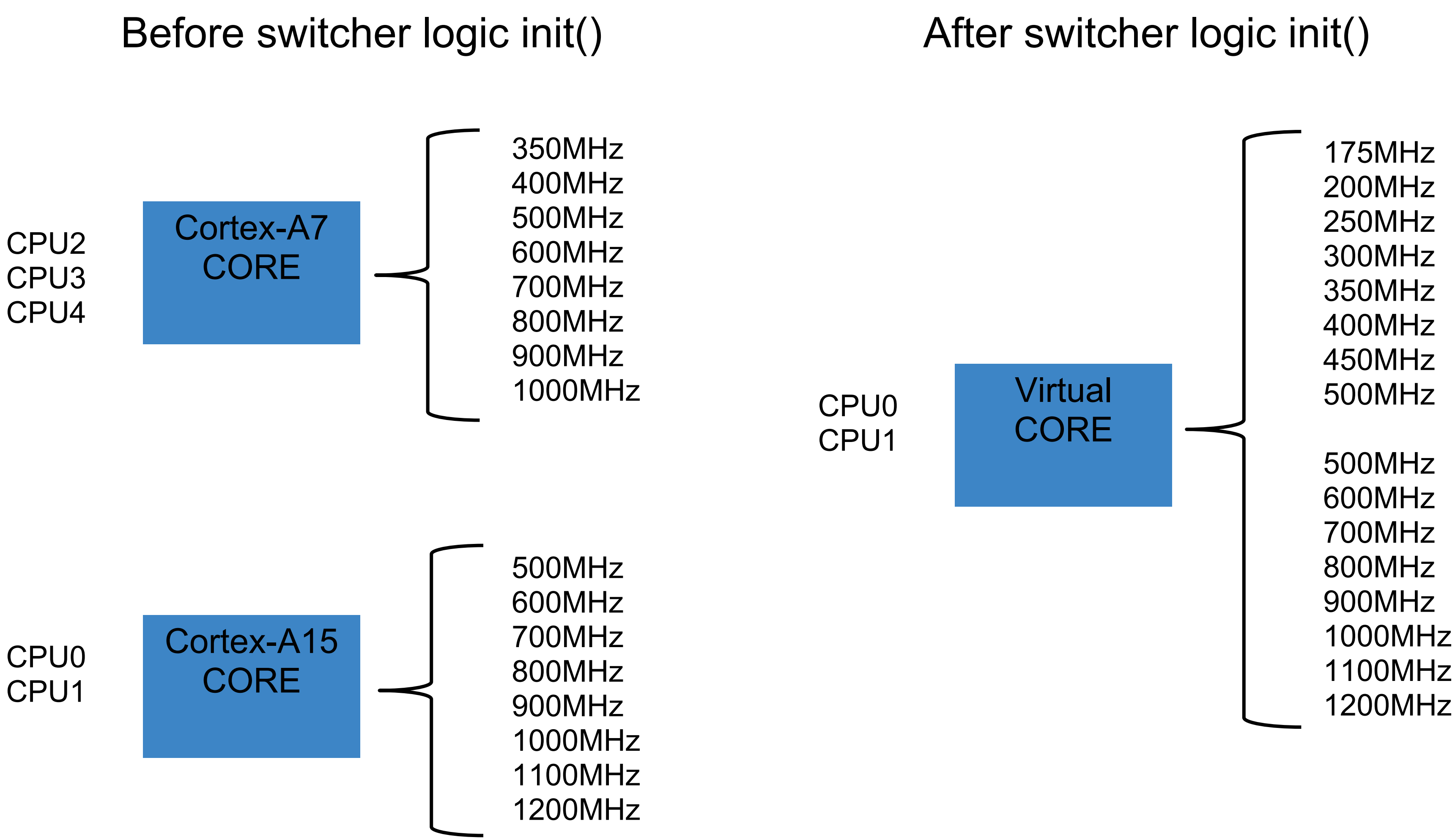
- A7 and A15 CPUs are physically numbered in each cluster:
  - A15_0, A15_1
  - A7_0, A7_1
- CPUs with a corresponding counterpart are grouped together:
  - {A15_0, A7_0}
  - {A15_1, A7_1}
- One CPU in each group is switched off:
  - A7_0, A7_1.

- Only the switcher needs to know about the grouping.

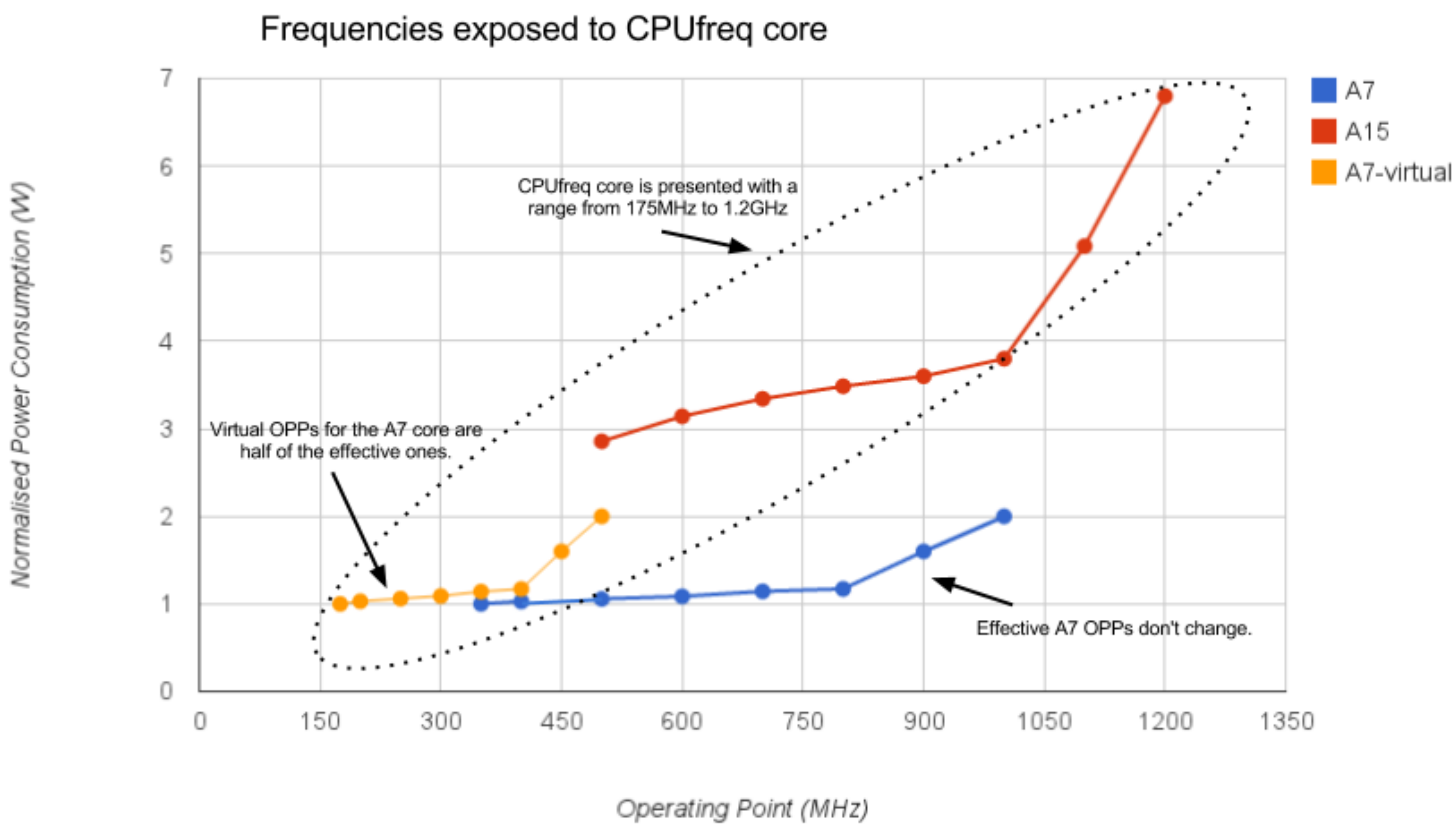
IKS - CPUfreq driver initialisation

- The cpufreq driver deals with the physical characteristic of each CPU core.
- Responsible of presenting the virtual CPUs' operating frequencies to the kernel.
- Select which core in a virtual CPU will be used.
- Also determines when to switch from one core to another in the "virtual" CPU.
- Switcher logic needs to be coordinated with cpufreq driver.

IKS - Frequencies exposed to CPUFREQ



IKS - Frequencies exposed to CPUFREQ



IKS - CPUfreq driver enhancement

- The CPUfreq core and the kernel are NOT aware of the b.L implementation.
- It is up to the CPUfreq driver to deal with the b.L architecture:

```
if (actual_cluster == A15_CLUSTER) AND
            (newFrequency < BIG_CLUSTER_MIN) {
        new_cluster = A7_CLUSTER;
} else if (actual_cluster == A7_CLUSTER) AND
            (newFrequency > SMALL_CLUSTER_MAX) {
        new_cluster = A15_CLUSTER;
}
...
...
...
if (actual_cluster != new_cluster)
        bL_switch_request(cpu, new_cluster);
```

IKS - Bridging the Chasm

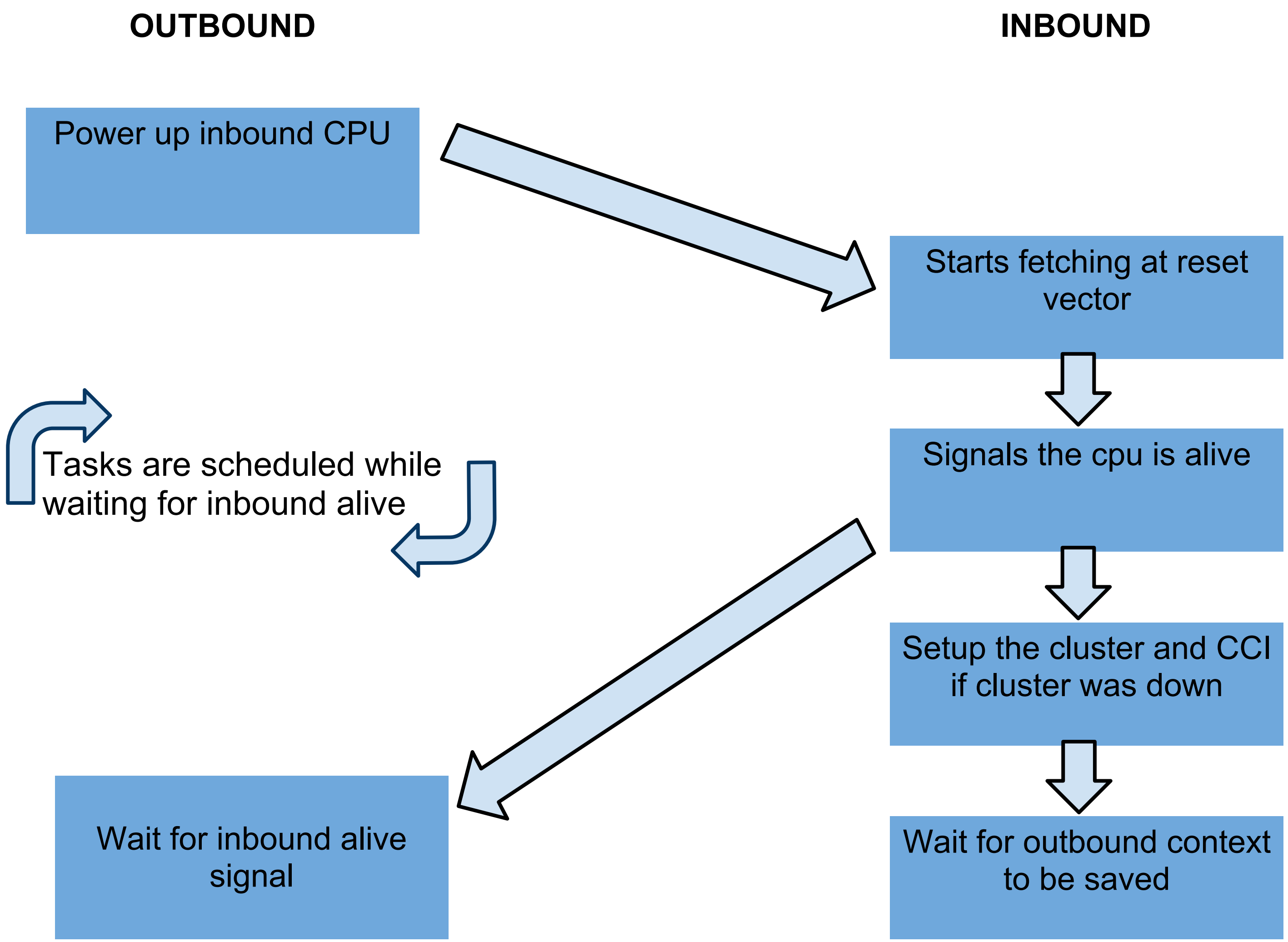
- Initial situation:
  - Virtual CPU0 is running a 200MHz.
  - Therefore A7_0 is active, A15_0 is switched off.
  - CPUfreq core knows CPU0 can go up to 1.2GHz.

- A request from the interactive governor comes in to go up to 1.0GHz.
- The A7 can't accommodate the request but the A15 can.

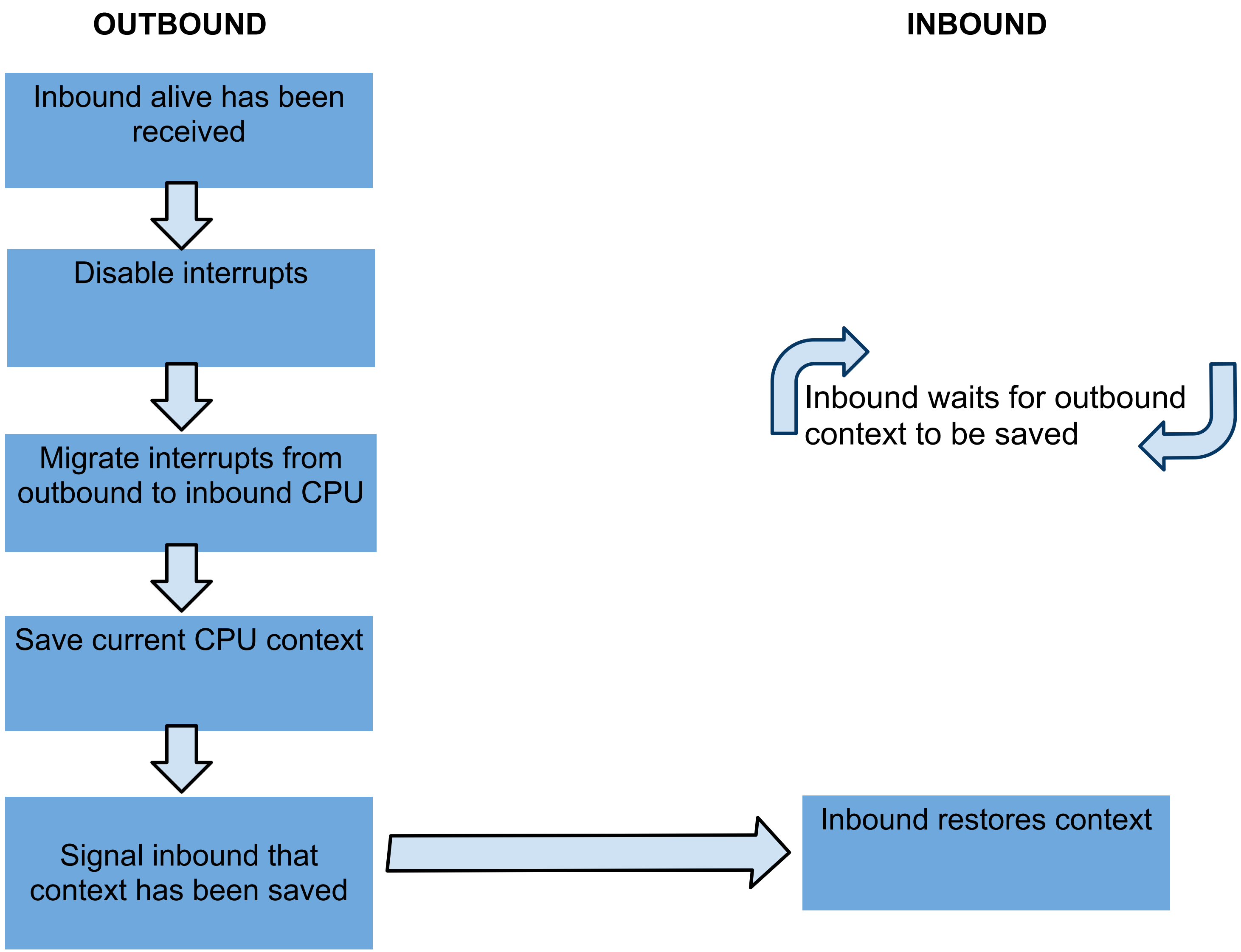
- What happens ?

- The CPUfreq driver instruct the switcher logic to move from the A7_0 (outbound) to the A15_0 (inbound).

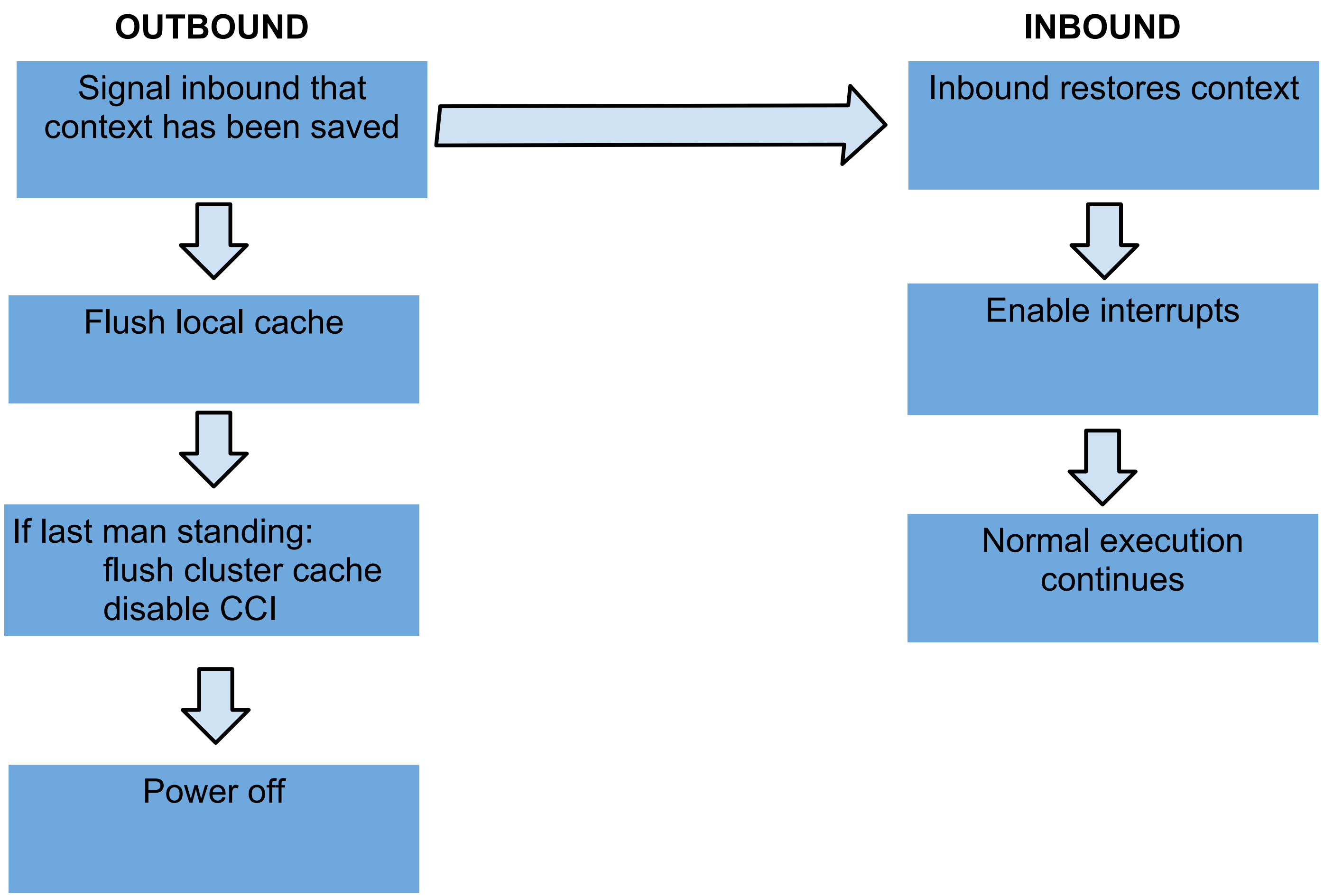
IKS - Bridging the Chasm



IKS - Bridging the Chasm



IKS - Bridging the Chasm



IKS - Bridging the Chasm

- Important things we haven't mentionned:
  - Mutual exclusion when setting up clusters (vlocks).
  - The last man standing algorithm.
  - The early poke mechanism.
  - CPU and cluster state tracking.

IKS - Addition to the Interactive Governor

- Though generic in nature and not tied to a distribution the IKS solution was tested using Android and the interactive governor.
- In it's original form the interactive governor algorithm reacts to the system load:
  - When the system is busy, it jumps to higher frequencies.
  - Above a certain threshold, moving from one OPP to another is further delayed by a timer.
- Since we have two cores in one virtual CPU, we duplicated the above algorithm to avoid reaching the overdrive (and costliest) point on the A15.

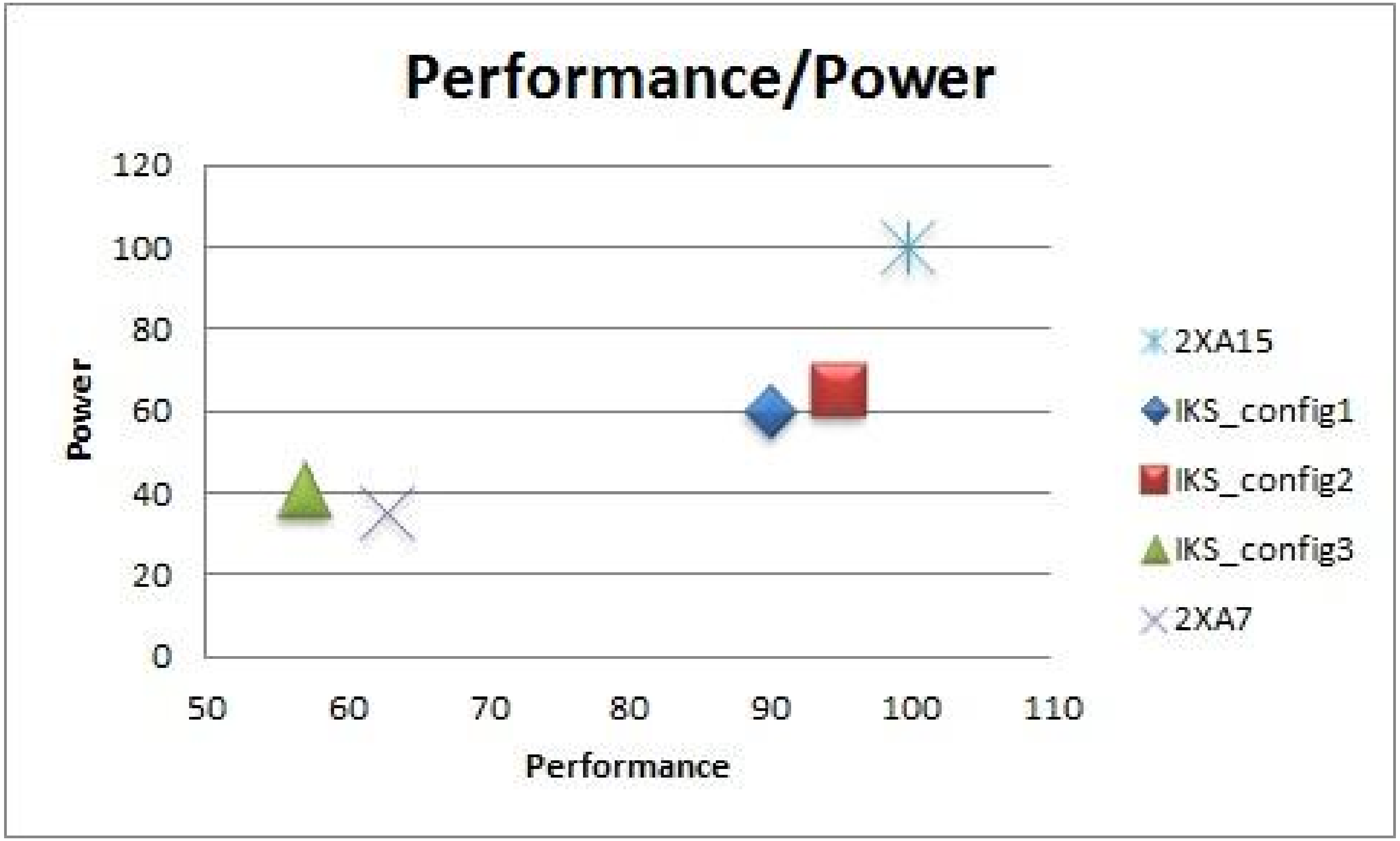
IKS - Addition to the Interactive Governor

IKS - The Results

- Our metrics:
  - Power consumed by each core.
  - BBench's "performance" metric, which gives a score for how fast web pages are loaded.
- Our test:
  - Running BBench with audio playing in the background.

- For IKS our goal was to obtain a 60/90 ratio:
  - 60% of the power used by a 2 x A15 solution.
  - 90% of the performance used by a 2 x A15 solution.

IKS - The Results



IKS - Tuning and Optimisation

- Basic configuration:
  - hispeed_freq = Max OPP on A7 = 500MHz
  - hispeed_freq2 = Last OPP on A15 before OD = 1GHz
  - hispeed_load = 85
  - hispeed2_load = 95

- Processing is done on the A7 cluster for as long as the CPU load is below 85%.
- When CPU load is between 85% and 95%, A15 core is used.
- When load goes above 95%, over drive frequencies on A15 (1.1GHz, 1.2GHz) are reached.

IKS - Tuning and Optimisation

- Optimisation was done using interactive governor.

- "above_hispeed_delay": the lower the value, the more responsive the system is.

- "timer_rate": how often the system is checked for frequency optimisation.

- Both are tightly coupled. Ex: if timer_rate is bigger than "above_hispeed_delay", opportunity for frequency adjustment will be lost.

IKS - Tuning and Optimisation

- IKS_config1:
  - above_hispeed_delay: 50ms
  - timer_rate: 10ms
  - Result: 60/90
- IKS_config2:
  - above_hispeed_delay: 0.5ms
  - timer_rate: 0.5ms
  - result: 65/95
- IKS_config3:
  - above_hispeed_delay: 750ms
  - timer_rate: 10ms
  - result:41/57

IKS - Upstreaming

- Cluster power management is being reviewed on the ARM Linux mailing list and getting positive remarks.
- All the source will be made public when one of our members has a release that utilises this code.

Question and Comments ?

Contributors to this project:

Nicolas Pitre
Dave Martin
Viresh Kumar
Mathieu Poirier
Amit Kucheria
David Zinman
Vishal Bhoj
Naresh Kamboju
Ryan Harkin
John (Tixy) Medhurst
Sudeep KarkadaNagesha
Achin Gupta
Robin Randhawa
Steve Bannister
Charles Garcia-Tobin
Ashok Bhat