Lecture 5

Big.LITTLE many-core CPU scheduling and load balancing

The same problem, a better solution

- With mobile devices and systems, the question is how to use the existing many cores in the best possible way, *saving energy* whenever possible, but also fully exploiting their computing power to *reduce the execution time* of applications.

- Computing devices have, most of the time, a low load and therefore they don’t need to keep all cores running.
- One package – two cores can be used when the load is low, and if there are more packages, the others can be asleep.
- Moreover, more energy can be saved if the two active cores are working at a low frequency. Can we use a more effective approach?
- CPU designers created two types of cores, *little* and *big*.

Heterogeneous many-cores

- While Intel and AMD use 4/8 *identical* cores, ARM uses cores of two different designs, e.g., a cluster of Cortex-A72 (high performance core) and a cluster of Cortex-A53 (energy efficient core). This configuration is known as the big.LITTLE architecture, where big cores (A72) work along with little ones (A53).

- The model: the scheduler starts by using little cores and when the workload increases, it starts to use big cores. Little cores go to sleep, until the workload starts to decrease when they are awaken.

- Strategy: for the big.LITTLE configuration, the scheduler picks the right core for the right load.

Case study: five smartphones

Model	Xperia M	Nexus 5X	Samsung A7	Galaxy S6	Samsung A5
Arch	2 same cores	4 L + 2 B cores	6 L + 2 B cores	4 L + 4 B cores	8 same cores
OS	Android 4.2	Android 6.0	Android 8.0	Android 7.0	Android 8.0
Kernel	Linux 3.4.0	Linux 3.18.10	Linux 4.9.44	Linux 3.18.48	Linux 4.9.44
Frequency GHz	1.0	1.4 / 1.8	1.6 / 2.2	1.5 / 2.1	1.9
Battery mAh	1750	2700	3300	2550	3000

L means little core
B means big core

Scheduling strategies

1. CPU migration: the scheduler assigns processes to a pair of cores. Each pair can have a little core and a big core, or two identical cores. The workload is moved inside the pair from a core to the other one. Both cores, or only one core of the pair can be active at a time. For example, for an octo-core smartphone, only four cores can be active at the same time.

- Example:
  - Sony Xperia M has two identical cores. As they are on the same package, the scheduler tries to balance them all the time. The main objective of the scheduler is to save energy, while reducing the execution time by increasing the cores' load.

2. Cluster migration

2. Cluster migration: cores are grouped in two clusters (domains) of two/four cores. The workload is moved between clusters.

- Examples:
  - *Nexus 5X:* the two big cores start the work and after few seconds their load goes to 0%; the four little cores load is increasing potentially up to 100%. The average load of big cores is around 20%. Explanation: saving energy is the objective.

  - *Galaxy S6*: a cluster of 4 little cores and one of 4 big cores.
  - Based on the workload, the system will decide to use little cores or big cores. Therefore, it follows the big.LITTLE strategy.

  - *Samsung A5*: 8 identical cores organized in two clusters.
  - Only one cluster of cores can be active at a time.
  - 1st core is always less loaded.

Global task scheduling

3. *Global scheduling*: a process can be executed by any core. The load is managed across all cores.

- Example:
  - *Samsung A7* has 6 little cores, ARM Cortex-A53 (8-stage pipeline) and 2 big cores, ARM Cortex-A73 (12-stage pipeline).
  - It follows exactly the big.LITTLE strategy, the big cores doing most of the heavy computation.
  - Cores of the same package are balanced against each other.
  - As all cores are used, the main objective is reducing the execution time.

Comparison

Objective	Xperia M	Nexus 5X	Samsung A7	Galaxy S6	Samsung A5
Min execution time			X	X	
Energy saving	X	X			X

App1 (300)	Xperia M	Nexus 5X	Samsung A7	Galaxy S6	Samsung A5
Execution time	1533 sec	257 sec	125 sec	94 sec	148 sec
Average load	92.5 %	65 %	56 %	53 %	78 %
Energy	15 %	2 %	1 %	1 %	< 1 %

Smartphones are using different scheduling and load balancing strategies, based on the cores configuration and on which objective is prioritized by the manufacturer.

Apple MacBook Air M1/2

- M1 is the CPU chip designed by Apple using 5nm technology – 16 billion transistors on chip!
- The new System-on-chip used by Apple has a CPU with 4 little cores and 4 big cores. Additionally, there are 7 or 8 GPU cores.
- Total power consumption is 10 W, a little core consuming 1/10 of the big core power consumption.

- M2 has up to 8 big cores and 4 little cores, which makes it 20% faster than M1. Its GPU can be configured with up to 19 cores.

Questions

1. Why would big cores be used immediately after the mobile device is turned on? Give one or more reasons and comment them.
2. What kind of applications can benefit of clusters of same cores?
3. How can a mobile mail app use little and big cores?

Conclusions

- Using little and big cores allows for significant power saving. Generally, demanding applications are executed by big cores.
- The big.LITTLE concept is not implemented according to the theoretical model by all manufacturers. There are differences the users should be aware of.
- Cores of the same type form clusters (domains) that can be managed as computing units. A cluster is similar to a virtual architecture.
- The big.LITTLE architecture is another innovation used to save energy of mobile devices.

References

- http://www.intel.com/technology/itj/2007/v11i4/9-process/2-intro.htm
- http://lwn.net/Articles/80911/
- https://www.androidauthority.com/fact-or-fiction-android-apps-only-use-one-cpu-core-610352/
- M. Pilaudeau, D.Grigoras: Experimental results regarding the workload of many-core mobile devices, ISPDC 2020, Warsaw, 5-8 July 2020, available on researchgate.org at
- https://www.researchgate.net/publication/342819849_Experimental_results_regarding_the_workload_of_many-core_mobile_devices