Micro-architectural Adaptation of Codelets using gem5 and CERE

Nicolas Derumigny
ENS de Lyon

Pablo de Oliveira Castro
Université de Versailles Saint-Quentin-en-Yvelines

Abstract

More and more efforts are deployed in order to increase the speed and the efficiency of CPUs. Therefore, these progresses target an average speeding ; whereas the scientific needs of computational power are specific. An architectural adaptation would be a great step forward in the power/performance ratio.

Sometimes the different regions of an application fits better on some architecture, depending of the type of data and the operations they are dealing with. The codelet approach, based on the use of the software Codelet Extractor and REplayer (CERE), allows to isolate and tune each section of the application individually, showing which improvements on the hardware side leads to better performance.

This report presents an overview of the possibility of micro-architectural adaptation using the x86 instruction set simulated by gem5, based on an performance-consumption ration, calculated with MCPAT. Its impact has been measured on both sequential and parallel applications on one CPU core, using the NAS and PARSEC benchmark suites.

1 Introduction

The development of new CPU architectures is a compromise between several parameters. The common approach is to minimise the cost and maximise the average execution time, measured on a suite of benchmark representative of the user's most frequent tasks. In HPC,the needs depend mostly of the application run: thus, architectural adaptations could greatly improve the power-efficiency of this application.

This adaptation is already in use on the embdded domain, with heterogeneus multicore systems (big.LITTLE) designed to decrease power consumption on small tasks (little cluster). They are nevertheless still able to deliver high performance on a small amount of time, thanks to the big cluster. This may be adapted on different calcul specialisation, as GPGPUs and CPUs already do: GPGPU are efficient in highly-scalable parallel applications, and CPUs are faster on sequential ones.

One other illustration of this material-to-software adaptation is the boost technologie, either on GPU than on CPU. It increases the frequency during high loads, while the chip is under a given temperature, or does not reach a thermal threshold.

The gem5 simulator[4] has been used to quantify micro-architecture impacts on softwares: it allows to fine-tune parameters without using different processors. As it emulates a whole linux system, the measurement could be really slow, that is why running only codelets gives a serious advantage comparing to measure the entire application. Comparing to running the IS full application in SE mode, running the IS codelet is four times faster.

Four x86 CPUs were simulated, one based on the Cortex A-15[8], the i5-3550 (with turbo and non-turbo frequency), the i5-3770U, a low-power mobile CPU and a QX9100. All these CPUs are simulated as one-core CPU, using the one-core values for non-shared caches and real value for shared caches.

The codelets used were extracted using CERE[7] tool, originally taken from NAS[2] sequential benchmark suite and PARSEC[3] benchmark suite. We chose NAS IS sequential as a simple serial application and pthread-disabled x264 for a more complicated one. Blackscholes and Freqmine, two OpenMP applications from PARSEC suite, were chosen to test multithread performance.

The energy consumption was calculate using MCPAT[9] and taken as a measure of the efficiency of each simulated CPU. Indeed, power consumption is the ideal measurement for the architecture tuning, as it delimit on one side the maximum computational power of the CPU at fixed architecture (due to frequency limits) and on the other side the cost to run it.

This paper explains in section 2 the related work and its line comparing to the current research. Section 3 describes the compatibility between gem5 and the codelets extracted by CERE, along with the models, the values and the applications chosen for the simulations. The results are commented on section 4. We finally conclude in the section 5.

2 Background

To our knowledge, no other work has been produced before using codelets to do architectural-tuning inside a simulaor.

2.1 On the benchmarking of processors

Moreover, benchmarks are a good way to reproduce the usage of a computer[5].

2.2 On the use of codelets

Source code isolation has already been validated as a reliable way to reproduce the comportement of applications. The in vivo code has to be replayed in vitro by extracting it and creating an application called codelet[1]. The codelet can therefore reproduce the comportement of the application without running a full benchmark. CERE is a sofware that extracts codelets from a C/C++/Fortran application using the LLVM compiler. It operates at the Itermediare Represemtation (IR) level, and thus is more flexible than code isolation or assembly isolation[7].

At this time, CERE targets *for()* loops and openMP parallel *for* loops. On sequential NAS IS benchmark, the selected codelet covers more than 98% of the total execution time, but is CHECK THE SPEEDUP times faster.

CERE captured the memory context at a page-granularity level, which is lighter than a full dump and then faster to replace in memory when replaying inside a simulator. Moreover, a cache warm-up is done before replaying the codelet by running one time the selected loop. This step should not be avoided when tuning microarchitecture parameters such as cache size or cache line size, as the warm-up could be slower but the other executions much faster.

2.3 On the simulators

The gem5 simulator

The gem5 simulator is an cycle-accurate simulator. Its accuracy has been proved on ARM simulation on both in-order and out-of-order processor, comparing real and simulated Cortex-A8 and Cortex-A9[8]. This comparisons reveals an average absolute error of only 7%.

Replaying SPLASH benchmark on gem5 shows an error on the execution time from 1.39% to 17.94%, explained by an inaccurate simulation of the DDR memory. Nevertheless, the gem5 simulator now handles different memory types, including modern DDR3, DDR4 and GDDR5, which should be more accurate than the tested DDR memory used in SPLASH tests[6].

The MCPAT simulator

3 Simulation framework

3.1 The gem5 simulator

The gem5 simulator can be run in two different modes: syscall emulation (SE) and fullsystem mode (FS). The Syscall emulation mode simulate only the comportement of the CPU inside a linux operating system, and therefore cannot efficiently simulate multi-threaded application, as no scheduler has been implemented. Besides, SE mode required a static linkage of all the required libraries.

On the contrary, the fullsystem mode emulate a full CPU; as the OS is emulated, the simulation is really slow (about fifteen minutes to boot linux on an x86 AtomicSimpleCPU). Nevertheless, FS mode is more accurate and more flexible. Indeed, FS mode can handle dynamic libraries, assuming that they are well installed in the virtual disk image. Moreover, gem5 featured a checkpoint functionality which avoid booting again when the CPU is changed.

3.1.1 Syscall emulation mode

Two changes on gem5 has been made to allow the use of CERE sequential codelets. First, the *getdents* syscall has been implemented, which is called inside the *readdir* function, used in the codelet memory mapping function. The second change concern a bug occurring when reading EOF with the syscall *read* while providing an invalid pointer: it should work and write nothing, but caused a page fault in gem5.

Both patch were submitted to gem5 community and wait for acceptation.

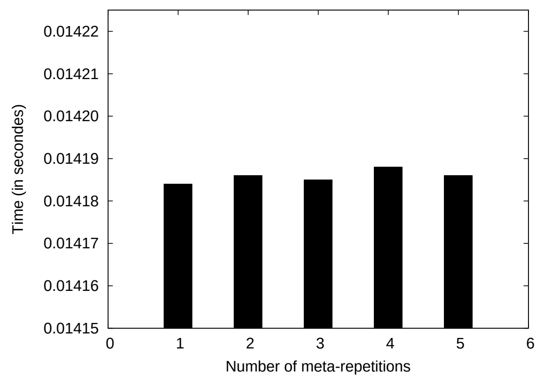

Figure 1: Variations of the codelet loop execution time on NAS IS class W benchmark using the i5-3550 configuration without turbo, with four CPUs.

In order to statically compile codelets with CERE[1], the argument *-static* must be provide at the linkage, either in the makefile or in the *lel.py* CERE source file. Noting that this could provoque some relocation error when specifying the entry point of the program: offset 0x60000000 (used as default start point in CERE) is used to place the standard C library in sequential NAS IS. Experimentally, offset 0x40000000 did not cause any trouble when compiling sequential codelets.

OpenMP parallel codelets cannot be run on SE mode due to the lack of real pthread implementation in the SE subsystem. Indeed, statically linking with *libiomp.a* results in a forced exit because of the unimplementation of the pthread management syscalls.

With these implementations, and assuming that all the syscalls have been implemented inside the gem5 simulator, any sequential codelet should work in SE mode. Given that there is no scheduler in gem5 SE mode, and given that there is no proper implementation of pthread implementation in SE mode[2], parallel codelets cannot be realistically replayed on SE mode.

Using the region __cere__is_ranked_475 with five meta-repetitions (six total runs of the loop, as one is used to warm the cache up), we observed 4x speedup, comparing to running the full benchmark[3]. All the data analysis is done on the median of these five meta-repetitions, but as the fluctuation is at most 0,03% (figure 1)[4], we can imagine running a codelet with only two or three meta-repetitions without significant biais.

			L2		
Name	Frequency[5]	L1D and L1I associativity[6]	Size	Assoc.	L3
Cortex-A15	1 GHz	2	1 MB	16	No
i5-3550	3,3-3,7 GHz	8	4×256 kB	8	Yes[7]
i5-3337U	1,8-2,7[8] GHz	8[9]	4×256	16	Yes[10]
Q9100	2,26 GHz	8	8 MB[11]	16	No

Figure 2: Parameters used for CPU simulations.

[1]CERE version 0.2 was used in this paper.

[2]M5thread has not been tested due the lack of scheduler and the miss of important syscall implementations in SE mode.

[3]Class W inputs were used for all the results.

[4]This is due to the deterministic routine of the codelet, as the region __cere__is_ranked_475 is verifying that a give array is well-sorted. Such tight results are harder to get on randomised or multithreaded code, see section 3.1.2.

[5]Non-turbo - Turbo frequency when turbo techology is implemented

[6]The L1D and L1I size is always 32 kB.

[7]The size of the L3 cache is set to 8MB and its associativity to 16-way due to gem5 limitations, it should be 6MB and 12-way.

[8]Only the non-turbo frequency has been simulated

[9]It should be 6 MB.

[10]It should be 3 MB.

[11]It should be 6 MB and 12-way.

3.1.2 Fullsystem mode

To run codelets in FS mode, only a few changes have been done: a more recent image than the ubuntu 7.04 available on gem5 site has been used, base on ubuntu-core 14.04. The kernel used is version 3.2.40 with default gem5 configuration.

To run OpenMP applications, just putting the *libiomp5.so* and *libomp.so* in */usr/lib*[12] works, excepting KMP_affinity which could not have been set to scatter to stabilize results. As a consequence, codelets on small inputs shows inexploitable comportements (figure 3) on four cores. Moreover, gem5 seems to hang when simulating more than one x86 CPU in FS mode, that is why all the applications (including multicore's one) were run using a one-core configuration.

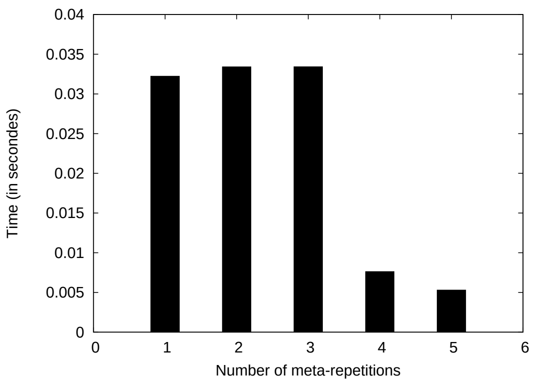

Figure 3: Variations of the codelet loop execution time on PARSEC Freqmine benchmark using the Cortex-A15 configuration with four AtomicSimpleCPU and simtest input.

3.2 Simulation models

3.2.1 Hardware configuration

The chosen configurations are detailed in figure 2. All systems are set with 8 GB of 1600 MHz DDR3, the cacheline size is always kept at 64 B, and all CPUs are quad-core without hyperthreading.

3.2.2 Chosen codelets

We chose three codelets to efficiently reproduce different usages:

- NAS IS sequential: region __cere__is_ranked_475 which check whether the calculated array is sorted or not.
- PARSEC[13] blackscholes: region __cere__blackscholes_m4__Z9bs_threadPv_first, an OpenMP region calculating the option value based on the Black & Scholes's equation.
- PARSEC freqmine: region __cere__tree8scan1_DBEP4Data_first which generate a hash from tree the dataset.
- PARSEC x264: region __cere__encoder_analyse_block_residual_write-_cabac_745 used in the CABAC encoding of the video.

4 Results

The power consumption of each CPU is calculated roughly using gem5 output file: such value should not be taken absolutly but relatively to other CPU simulated. One have to keep in ,ind that the CPUs use only using a generic x86 scheme, without hyper-threading or other technological improvements (pipeline aside), and own only one core. That is why only relative measured at fixed codelets are really meaningfull.

4.1 Power/performance ratio and index

The performance of a CPU is measured by the execution time of the selected codelet. To keep an higher-is-better index, only $1/t_e$ values are used (where t_e is the execution time). The power-comsumpution ration is defined as

$$\frac{1}{P.t_e} \tag{1}$$

With P the power consumption of the CPU on the benchmark.

A higher ratio means either better performance or less comsumption, and so a better choice.

[12]Taken from linux mint MATE 17.3 64 bits
[13]Version 3.0-beta-20150206

4.2 Performance

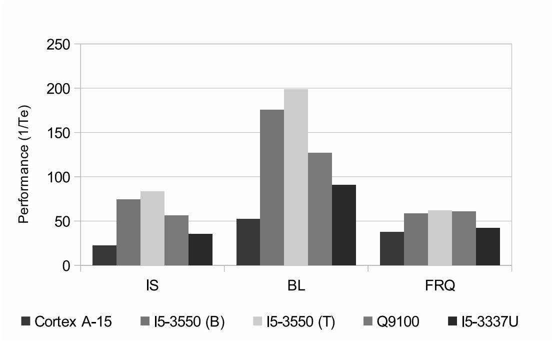

Figure 4: Performance of all tested CPUs on differents codelets.

X264 codelet is too fast too measure the execution time with gem5 (only 0,0000001 or 0,0000002 seconds are calculated), so this codelet will only be analyse for its power consumption.

IS and freqmine does execute on the same order of time (figure 4), even if freqmine codelet was extracted running the simtest input.

4.3 Power consumption

When the power consumption is on the x axis, the former is graduated backward to keep and higher-is-better index.

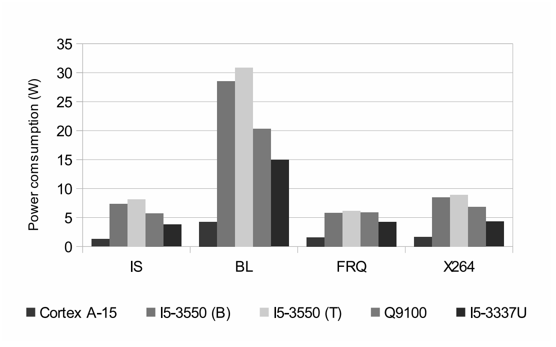

Figure 5: Performance-Power consumption ratio for each CPU.

For example, the blackscholes codelets seems to consume more power than all the other codelets (figure 5). This is due to floating point operations, used significantly only in this codelet (see 4.3.2).

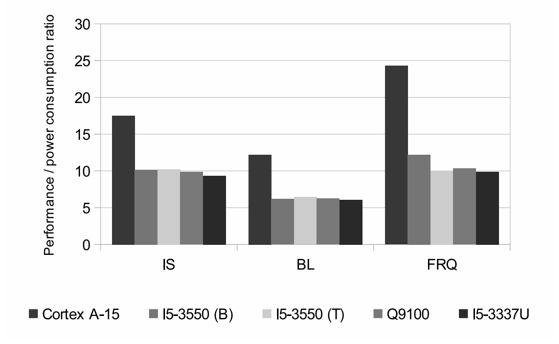

Figure 6: Performance-Power consumotion ration for each codelets.

4.3.1 IS : A serial applications

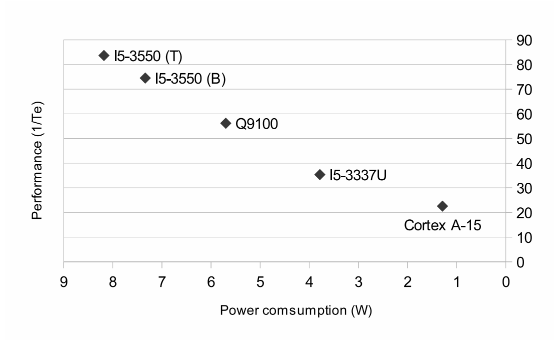

Figure 7: Power-Performance graph for the IS codelet.

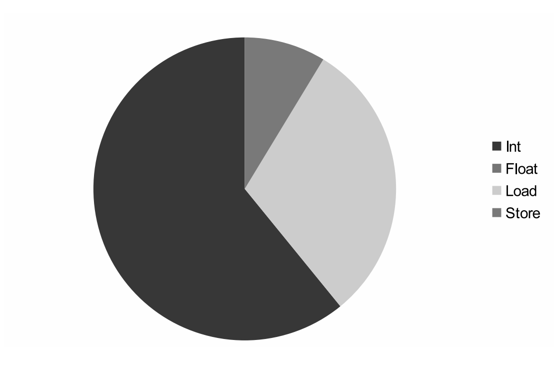

Figure 8: Repartition of the instructions during the execution of the IS codelet.

x264

?

4.3.2 Parallel applications

This parallel applications are run on a one-core configuration: these results are only bound to show the single-core performance-consumption differences, and not the manycore scaling. Such studys could easily be measured on ARM systems (see 6.1).

Freqmine

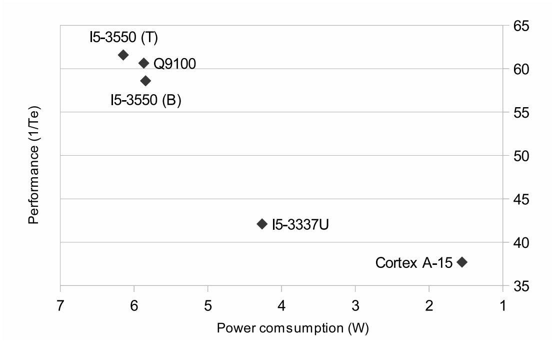

Figure 9: Power-Performance graph for the Freqmine codelet.

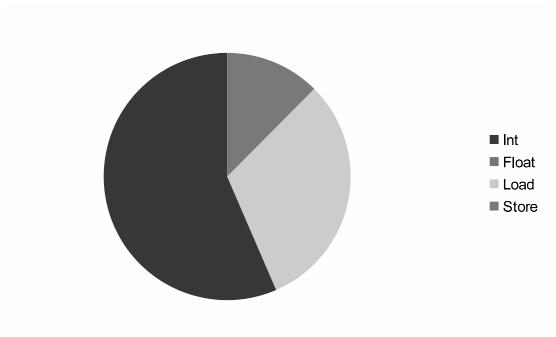

Figure 10: Repartition of the instructions during the execution of the Freqmine codelet.

Blackscholes

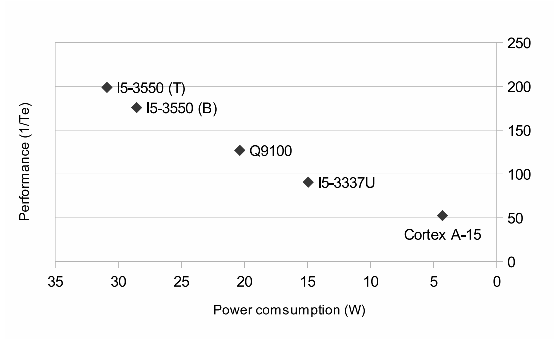

Figure 11: Power-Performance graph for the Blackscholes codelet.

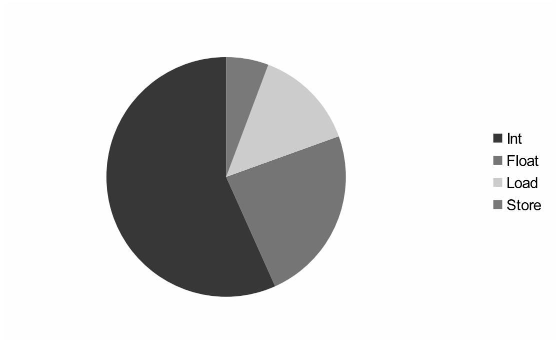

Figure 12: Repartition of the instructions during the execution of the Blackscholes codelet.

4.4 Advantages

The use of codelets gives a powerfull utility to fine-tune the architecture fitting to the codelets. Because of CERE limitations, only codelets replay *for* loops or *omp parallel for* loops are supported, but this is much lighter than running the full application. Moreover, codelets can improve individually each region of an application, which is extremely usefull to choose on which CPU cluster run the application on a big.LITTLE system, or even in an heterogeneus compute server.

The gem5 simulator is a precise tool to reproduce the comportement of a specific machine on a general computer. As the simulation is quite slow, the use of codelets could be really usefull in term of time and ressources in the conception of heterogeneus servers. As CERE operates as an IR-level, the operation does not need any additionnal work on the source code, and can be run in C, C++ or Fortran programs. Besides, the compilation of the codelets uses the power of the host machine and not the simulated one, which is really usefull in term of compute time: the emulated system is used only when it is truely needed.

4.5 Inconvenient : Unknown precision

Gem5 is a simulator, and therefore cannot be entirely trusted. CERE too cannot exactly replay a whole execution of ann application ; and the measures output by MCPAT are only an estimation of the effective power consumption of the CPUs: that's why the results could not fit exactly to the real-life experiments. Nevertheless, these simulations are bound to give an overview of the gains that could be archived by improving a specific part of a CPU, not to give absolute results.

The gem5 simulator has not been validated yet on x86 accuracy, and could be quite biased, as it emulates a generic x86 processor, whereas state of the art CPUs are even more complex.

5 Conclusion

References

[1] Chadi Akel, Yuriy Kashnikov, Pablo de Oliveira Castro, and William Jalby. Is source-code isolation viable for performance characterization? In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*, pages 977–984. IEEE Computer Society, 2013.

[2] David H. Bailey, Eric Barszcz, John T. Barton, D. S. Browning, Robert L. Carter, Leonardo Dagum, Rod A. Fatoohi, Paul O. Frederickson, T. A. Lasinski, Robert Schreiber, Horst D. Simon, V. Venkatakrishnan, and Sisira Weeratunga. The nas parallel benchmarks. *IJHPCA*, 5(3):63–73, 1991.

[3] Christian Bienia. *Benchmarking Modern Multiprocessors.* PhD thesis, Princeton University, January 2011.

[4] Nathan L. Binkert, Bradford M. Beckmann, Gabriel Black, Steven K. Reinhardt, Ali G. Saidi, Arkaprava Basu, Joel Hestness, Derek Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[5] Maximilien Breughe and Lieven Eeckhout. Selecting representative benchmark inputs for exploring microprocessor design spaces. *TACO*, 10(4):37, 2013.

[6] Anastasiia Butko, Rafael Garibotti, Luciano Ost, and Gilles Sassatelli. Accuracy evaluation of GEM5 simulator system. In Leandro Soares Indrusiak, Guy Gogniat, and Nikolaos S. Voros, editors, *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), York, United Kingdom, July 9-11, 2012*, pages 1–7. IEEE, 2012.

[7] Pablo de Oliveira Castro, Chadi Akel, Eric Petit, Mihail Popov, and William Jalby. CERE: llvm-based codelet extractor and replayer for piecewise benchmarking and optimization. *TACO*, 12(1):6, 2015.

[8] Fernando A. Endo, Damien Couroussé, and Henri-Pierre Charles. Micro-architectural simulation of in-order and out-of-order ARM microprocessors with gem5. In *XIVth International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2014, Agios Konstantinos, Samos, Greece, July 14-17, 2014*, pages 266–273. IEEE, 2014.

[9] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In David H. Albonesi, Margaret Martonosi, David I. August, and José F. Martínez, editors, *42st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42 2009), December 12-16, 2009, New York, New York, USA*, pages 469–480. ACM, 2009.

[10] ARM ltd. Arm information center, 2015.

6 Annexe

6.1 ARM Simulation

As multicore benchmarks seem to hang on X86 simulation, some work has been done to adapt the codelet on ARM simulation. The multicore simulation works with linaro-minimal

As capturing on ARM required an ARM machine, a few cross-compile instructions has been added in CERE. Capturing en gem5 may indeed take several days, and require moreover an ARM version of CERE pre-installed on the virtual machine, which is too heavy to be implemented yet. The goal is then to use an x86 memory dump and replay it on ARM systems. The following changes have been made in CERE:

- Changed objdump to aarch64-linux-gnueabi-objdump.
- Added clang cross-compile option.

After thoses changed, the compilation ran successfully, but the execution outputs *"Killed"*, even before the main() starts.

We found that the aarch64 architecture limits the application space to the address 0x00000000_00000000 to 0x0000ffff_ffffffff[10]. But the stack is placed on x86 linux at address 0x07fda3c7_b0000000 so the kernel kills the application as soon as the X86 stack region is reserved. To avoid this issue, the stack has manually been moved to address 0x000003c7_b0000000.

NAS IS (codelet __cere__is_ranked_475) has been successfully replayed on a juno board (linaro-image-minimal-genericarmv8 system) using this trick. Nevertheless, further adaptations have to be done in order to safely convert x86 dumps to ARM dumps[14].

6.2 About the UVSQ laboratory

6.3 Gem5 bugs

[14] Especially on pointers of stack address which need to be updated to the new stack position at the codelet compilation.