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 to increase the speed and the efficiency of the CPUs. Therefore, these progresses target a general speeding ; whereas the diversity of the applications leads us to think of an architectural adaptation.

Sometimes, even the different regions of the application have different architecture preferences, 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 present an overview of the possibility of micro-architectural adaptation using the x86 instruction set simulated by gem5, based on an power-efficient point of view.Its impact has been measured on both sequential and parallel applications using the NAS and PARSEC benchmark suites.

1 Introduction

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

This adaptation is already in use on a smaller scale: heterogeneus multicore systems (big.LITTLE), widely spreads in the embedded domain, are designed to decrease power consumption on small tasks (little cluster). They nevertheless still be 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 adaptation is the boost technologie, either on GPU than on CPU. It increase frequency during high loads, while the chip is under a given temperature, or does not reach a thermal threshold.

The gem5[3] has been used to realistically quantify micro-architecture impacts on softwares: it allowsto fine-tune parameters without using different processors. As it emulate a whole linux system, the measure could be really slow, that is why running only codelets gives a serious advantage comparing to measure the entire application.

Four x86 CPUs were simulated, two based on the Cortex A-15 and the Cortex-A7 models[5], the i5-3550 and the x5-Z8300, both with turbo and non-turbo frequency. All of them are quad-core without hyperthreading, as it is often disable in clusters due to non-stable performance. Due to simulation time, all CPUs where simulated using the AtomicSimpleCPU.

The codelets used were extracted using CERE[4] tool, originally taken from NAS[1] sequential benchmark suite and PARSEC[2] 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 multicore performance.

The energy consumption was calculate using MCPAT[6] 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 4.2. We finally conclude in the section 5.

2 Background

TODO !

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 the 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.

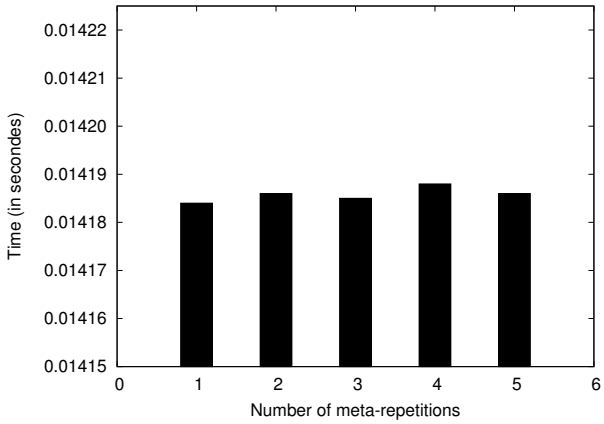

Figure 1: Variations of the codelet loop execution time on NAS IS class W benchmark using the i5-3550 configuration without turbo.

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.

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.

[1]Only the version 0.2 of CERE 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]IS class W was 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 when turbo techology is implemented

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

Name	Frequency[5]	L1D and L1I associativity[6]	L2		L3
			Size	Assoc.	
Cortex-A7	1,4 GHz	4	512 kB	8	No
Cortex-A15	1 GHz	2	1 MB	16	No
x5-Z8300	1,44-1,84 GHz	4	1 MB	16	No
i5-3550	3,3-3,7 GHz	8	4×256 kB	8	Yes[7]

Figure 2: Parameters used for CPU simulations.

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[8].

To run OpenMP applications, just putting the *libiomp5.so* and *libomp.so* in */usr/lib*[9] 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).

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[10] 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.

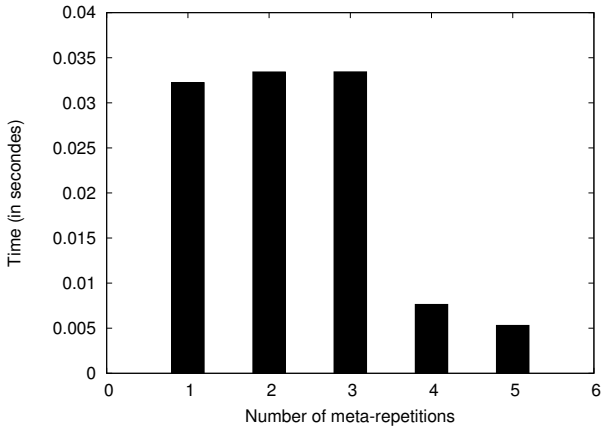

Figure 3: Variations of the codelet loop execution time on PARSEC Freqmine benchmark using the Cortex-A15 configuration with AtomicSimpleCPU and simtest input.

4 Results

4.1 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.

[7]The size of the l3 cache is set to 8MB and its associativity to 16-way due to gem5 limitation, it should be 6MB and 12-way.
[8]except the maximal number of cores set to 4.
[9]Taken from linux mint MATE 17.3 64 bits
[10]Version 3.0-beta-20150206

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.

4.2 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 efffective power consumption of the CPUs: that's why the results could not fit exactly to the real-life experiments. Nevertheless, these simulation are bound to give an overview of the gains that could be archived by improving a specific part of a CPU.

5 Conclusion

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.