Is Arm software ecosystem ready for HPC?

Research Poster – Extended Abstract

Fabio Francisco Banchelli Gracia
Barcelona Supercomputing Center
fabio.banchelli@bsc.es

Daniel Ruiz Muñoz
Barcelona Supercomputing Center
daniel.ruiz@bsc.es

Ying Hao Xu Lin
Barcelona Supercomputing Center
ying.xulin@bsc.es

Filippo Mantovani
Barcelona Supercomputing Center
filippo.mantovani@bsc.es

KEYWORDS

Arm, Arm HPC Compiler, Arm Performance Libraries, ATLAS, BLAS, OpenBLAs, WRF, FEniCS

ACM Reference format:
Fabio Francisco Banchelli Gracia, Daniel Ruiz Muñoz, Ying Hao Xu Lin, and Filippo Mantovani. 2017. Is Arm software ecosystem ready for HPC?. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, Colorado USA, November 2017 (SC17),* 3 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, the HPC community has increasingly grown its interest towards the Arm architecture with research projects targeting primarily the installation of Arm-based clusters. State of the art research project examples are the European Mont-Blanc, the Japanese Post-K, and the UKs GW4/EPSRC.

Primarily attention is usually given to hardware platforms, and the Arm HPC community is growing as the hardware is evolving towards HPC workloads via solutions borrowed from mobile market e.g., big.LITTLE and additions such as Armv8-A Scalable Vector Extension (SVE) technology. However the availability of a mature software ecosystem and the possibility of running large and complex HPC applications plays a key role in the consolidation process of a new technology, especially in a conservative market like HPC.

For this reason in this poster we present a preliminary evaluation of the Arm system software ecosystem, limited here to the Arm HPC Compiler and the Arm Performance Libraries, together with a porting and testing of three fairly complex HPC code suites: QuantumESPRESSO, WRF and FEniCS.

The selection of these codes has not been totally random: they have been in fact proposed as *HPC challenges* during the last two editions of the Student Cluster Competition at ISC [12] where all the authors have been involved operating an Arm-based cluster and awarded with the Fan Favorite award.

SC17, November 2017, Denver, Colorado USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

A state of the art status of the Arm system software ecosystem can be found in [10]: this worked as starting point for our work. Of course we have been strongly inspired by our previous work [9], but we differentiate from it because *i)* we focus on system software and applications and *ii)* we evaluate Arm-v8 platforms.

We also did not put any effort in tuning codes nor compilers in order to get more vectorized code like proposed in [4] because we wanted to emulate an end-user experience, taking his current code and executing *as-is* on an Arm-based platform.

Also, we did not extend our evaluation to novel Arm extensions, like e.g., the SVE study presented in [6, 11] because the level of maturity of SVE do not allow us to test on real platforms with real complex HPC codes.

3 METHODOLOGY

We structured our preliminary study of the HPC system software for Arm first evaluating the compilers technology available, then the most common math library solutions and finally using large production HPC codes. For this study we used some of the Arm-based *mini-clusters* installed at Barcelona Supercomputing Center within the Mont-Blanc project [5].

3.1 Compilers comparison

We perform a comparison between the GNU Compiler Collection (GCC) 7.1.0 and the Arm HPC Compiler 1.3.0. The Arm HPC Compiler has been developed by Arm for Arm-based platforms to be used in HPC environments and it is based on LLVM. C, C++ and Fortran compilers are included in both suites. As we evaluate them on a multi-core shared memory environment the OpenMP runtimes become also relevant. Both have been evaluated using the default OpenMP runtime: `libgomp` for GCC and `libomp` for the Arm HPC Compiler.

For the comparison in a serial / single-core environment we use the Polybench benchmark suite [14], while for the study

in a shared memory context, we use two well recognized mini-apps, Lulesh and CoMD from the ExMatEx effort [2]. In all cases we use the flags `-O3 -mcpu={cortex-a57,thunderx}`.

3.2 Math library comparison

BLAS, LAPACK and FFT are a collection of low-level math routines commonly used by the scientific community. The Arm Performance Libraries implement them and are maintained, supported and tuned by Arm for a wide range of Arm-based SoCs using OpenMP for their parallel version. We compared the following flavours of math libraries:

- Arm Performance Libraries 2.2.0
- ATLAS 3.11.39 + FFTW 3.3.6
- OpenBLAS 0.2.20 + FFTW 3.3.6.

Since neither ATLAS nor OpenBLAS include an FFT implementation, they must be combined with an FFT implementation, for which we choose FFTW.

For the evaluation we first consider a DGEMM micro-benchmark and then a large HPC production code, QuantumESPRESSO [8]. The latter is mostly written in Fortran and uses MPI and (optionally) OpenMP to exploit parallelism. As its implementation uses BLAS, LAPACK and FFT functions, it is a good candidate for the evaluation of the Arm Performance Libraries in a real HPC application. We evaluated the execution time of the MPI only code running the *pwscf-small* input set [7] on 8 cores of our AMD Seattle SoC [5]. For this evaluation we used GCC 7.1.0 to compile the codes.

3.3 Production HPC applications

Besides QuantumESPRESSO used for evaluating the math-library, we consider in our study two more applications: WRF [15] and FEniCS [13]. The common idea behind this choice is to try to evaluate the effort of porting, building and running two complex and modular codes relevant for scientists and for the HPC community. The second reason for choosing them is because both of them were part of the Student Cluster Competition challenges of 2016 and 2017 [12], in which all the authors of this poster have been involved.

For WRF we briefly present the porting effort needed for having it built on two Arm-based clusters and its scalability. For FEniCS we present as well the list of dependencies needed for having the suite up and running on an Arm-based computing environment and we show that, thanks to the careful recompilation of all the components, a real parallel execution on the FEniCS suite is possible.

Both WRF and FEniCS do not offer a native distribution that can be considered plug-and-play for Arm. For this reason we plan to release our porting effort in the official Arm repository of HPC application and system software [1].

3.4 Future work

As already mentioned, we plan to repeat our evaluation of the math libraries using the Arm Performance Libraries combined with FFTW, while waiting for a more optimized version of FFT functions from Arm.

For the compilers comparison, we plan to use clustering techniques to analyze the behaviour of number of total instructions vs instructions per cycles (IPC) in different parallel configurations with Lulesh and CoMD.

We also plan to extend our evaluation of WRF involving more input sets and more MPI ranks for the NVIDIA JetsonTX1 platform. Also, besides performance figures we will include *energy to solution* and *energy delay product* considerations.

4 CONCLUSIONS

Concerning compilers, in a serial environment we notice significant better performance of the Arm HPC Compiler in benchmarks involving matrix-matrix operations and on average we reproduce the already studied behaviour [3]. In a parallel environment, we noticed better performance at higher number of cores and fine grained parallelism using Arm HPC Compiler. This seems to be related to lower overhead of thread creation/destruction in the OpenMP runtime of the Arm HPC Compiler. In presence of a more coarse grain parallelism, the overhead of thread creation/destruction is less significant and there is a similar tendency of the two compilers with a slight advantage $\sim$20% in favour of GCC.

Concerning math-libraries, the performance obtained with the Arm Performance Libraries 2.2.0 in a DGEMM micro-benchmark is comparable with the one offered by other optimized libraries for small matrix sizes. Performance gain can be noticed with matrix sizes bigger than 1500×1500 elements. Arm Performance Libraries also suffer probably a premature implementation of the FFT-related functions that can be compensated for the moment calling functions of the FFTW library.

The evaluation of large production HPC codes has been flawless, allowing us to build and execute two modular suites, WRF and FEniCS, supporting all the available parallelism, without major obstacles.

In general, the Arm software ecosystem has been proven mature enough for running on Arm-based cluster without major restrictions even in an educational project like Student Cluster Competition. Performance wise, the results are in-line with other mainstream comparable software (e.g., GCC, ATLAS, BLAS, OpenBLAS) with two exceptions: *i)* the Arm tools allow to reach a significant better performance in matrix-matrix operations, while *ii)* further optimization work should be done for the FFT part.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] and Horizon 2020 under the Mont-Blanc projects [9], grant agreements n. 288777, 610402 and 671697. The authors would also like to thank *E4 Computer Engineering* for providing part of the hardware resources needed for the evaluation carried out in this poster as well as for greatly supporting the Student Cluster Competition team.

REFERENCES

[1] ARM HPC Users Group / packages 2017. (2017). Retrieved 30 Jul 2017 from https://gitlab.com/arm-hpc/packages/wikis/home

[2] Exascale Co-design Center for Materials in Extreme Environments 2017. (2017). Retrieved 30 Jul 2017 from http://www.exmatex.org

[3] GCC7 vs Clang4 comparison 2017. (2017). Retrieved 30 Jul 2017 from http://www.phoronix.com/scan.php?page=article&item=gcc7-clang4-jan&num=1

[4] G. Mitra, B. Johnston, A. P. Rendell, E. McCreath, and J. Zhou. 2013. Use of SIMD Vector Operations to Accelerate Application Code Performance on Low-Powered ARM and Intel Platforms. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*. 1107–1116. https://doi.org/10.1109/IPDPSW.2013.207

[5] Mont-Blanc mini-clusters user documentation 2017. (2017). Retrieved 30 Jul 2017 from https://wiki.hca.bsc.es/dokuwiki/mont-blanc_machines

[6] Porting and Adapting Dense and Sparse Matrix Application to ARM's SVE 2017. (2017). Retrieved 30 Jul 2017 from http://www.goingarm.com/slides/2017/GoingARM-UofM-2017.pdf

[7] Quantum ESPRESSO, input set 2017. (2017). Retrieved 30 Jul 2017 from http://qe-forge.org/gf/project/q-e/frs/?action=FrsReleaseBrowse&frs_package_id=36

[8] Quantum ESPRESSO, integrated suite of Open-Source computer codes 2017. (2017). Retrieved 30 Jul 2017 from http://www.quantum-espresso.org

[9] Nikola Rajovic, Alejandro Rico, Filippo Mantovani, Daniel Ruiz, Josep Oriol Vilarrubi, Constantino Gomez, Luna Backes, Diego Nieto, Harald Servat, Xavier Martorell, Jesus Labarta, Eduard Ayguade, Chris Adeniyi-Jones, Said Derradji, Herve Gloaguen, Piero Lanucara, Nico Sanna, Jean-Francois Mehaut, Kevin Pouget, Brice Videau, Eric Boyer, Momme Allalen, Axel Auweter, David Brayford, Daniele Tafani, Volker Weinberg, Dirk Brmmel, Ren Halver, Jan H. Meinke, Ramon Beivide, Mariano Benito, Enrique Vallejo, Mateo Valero, and Alex Ramirez. 2016. The Mont-blanc Prototype: An Alternative Approach for HPC Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Press, Piscataway, NJ, USA, 38:1–38:12. http://dl.acm.org/citation.cfm?id=3014904.3014955

[10] Alejandro Rico, Jos A. Joao, Chris Adeniyi-Jones, and Eric Van Hensbergen. 2017. ARM HPC Ecosystem and the Reemergence of Vectors: Invited Paper. In *Proceedings of the Computing Frontiers Conference (CF'17)*. ACM, New York, NY, USA, 329–334. https://doi.org/10.1145/3075564.3095086

[11] N. Stephens, S. Biles, M. Boettcher, J. Eapen, M. Eyole, G. Gabrielli, M. Horsnell, G. Magklis, A. Martinez, N. Premillieu, A. Reid, A. Rico, and P. Walker. 2017. The ARM Scalable Vector Extension. *IEEE Micro* 37, 2 (March 2017), 26–39. https://doi.org/10.1109/MM.2017.35

[12] Student Cluster Competition web site 2017. (2017). Retrieved 30 Jul 2017 from http://hpcadvisorycouncil.com/events/student-cluster-competition/

[13] The FEniCS computing platform 2017. (2017). Retrieved 30 Jul 2017 from https://fenicsproject.org/

[14] The Polyhedral Benchmark suite 2017. (2017). Retrieved 30 Jul 2017 from http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/

[15] The Weather Research & Forecasting Model 2017. (2017). Retrieved 30 Jul 2017 from http://www.wrf-model.org