Rapporti Tecnici INAF
INAF Technical Reports

INAF Technical Report: Evaluating SoC power efficiency through N-body application

D. Goz[a], S. Bertocco[a], I. Coretti[a], A. Ragagnin[a], L. Tornatore[a], G. Taffoni[a]

[a]*INAF-Osservatorio Astronomico di Trieste, Via G. Tiepolo 11, 34131 Trieste - Italy*

Abstract

Currently, the High Performance Computing (HPC) sector is undergoing a profound phase of innovation, in which the main stopper in order to achieve "exascale" performance is the power-consumption. The usage of "unconventional" low-cost computing systems is therefore of great interest for several scientific communities looking for a better trade-off between performance and power consumption. In this technical report, we make a performance assessment of commodity low-power System on Chip (SoC) using a direct N-body application for astrophysics. We also describe the methodology we have employed to measure the power drained by the application while running. We find that SoC technology could represent a valid alternative to traditional technology for HPC in terms of good trade-off between time-to-solution and energy-to-solution. This work arises in the framework of the `ExaNeSt` and `EuroExa` projects, which investigate the design of a SoC-based, low-power HPC architecture with a dedicated interconnection scalable to million of compute units.

Keywords: `ExaNeSt`, `EuroExa`, `SoC`, `ARM`, `GPU`, `power-efficiency`

Email addresses: `david.goz@inaf.it, ORCID:0000-0001-9808-2283` (D. Goz), `sara.bertocco@inaf.it, ORCID:0000-0003-2386-623X` (S. Bertocco), `igor.coretti@inaf.it, ORCID:0000-0001-9374-3249` (I. Coretti), `antonio.ragagnin@inaf.it, ORCID:0000-0002-8106-2742` (A. Ragagnin), `luca.tornatore@inaf.it, ORCID:0000-0003-1751-0130` (L. Tornatore), `giuliano.taffoni@inaf.it, ORCID:0000-0002-4211-6816` (G. Taffoni)

1. Introduction and motivation

New challenges in Astrophysics are urging the need of a large number of computationally intensive simulations. New High Performance Computing (HPC) facilities are mandatory to address the size of data coming from upcoming observational instruments and of the exceptionally computationally intensive simulations to interpret such observations and to make theoretical predictions. However, the main stopper in the achievement of "exascale" computational infrastructures is the power consumption.

Technology is rapidly shifting towards power-efficient Systems-on-Chip (SoC), designed to meet the requirements of the scientific community. These hardware platforms host integrated circuits composed of multicore CPUs combined with either graphic-processing units (GPUs) or Field Programmable Gate Arrays (FPGAs) aiming to optimize the energy-to-performance ratio.

Commonly, in order to benchmark the performance of platforms, test suites publicly available and documented in literature are used, namely the HPCG Benchmark[1] and LINPACK Benchmark[2], both metrics for ranking HPC systems. The results of such metrics are in terms of pure performance (time-to-solution), i.e. they are designed to perform computational, communication, and memory access patterns, without give any insight about the power-efficiency (energy-to-solution) of the devices exploited in the platforms.

In this technical report we choose a real-scientific application commonly used in the Astrophysics sector, instead of a synthetic benchmark, in order to assess the power-efficiency of SoC architecture. The following research is related with the activity done by INAF in the framework of the `ExaNeSt`[3] and `EuroExa`[4] projects. They are delivering prototypes based on low power-consumption ARM64 processors, accelerators and low-latency interconnections

[1] Available: https://www.hpcg-benchmark.org/

[2] Available: http://www.netlib.org/benchmark/hpl/

[3] https://exanest.eu/

[4] https://euroexa.eu/

implementing a co-design approach where scientific applications requirements are driving the hardware design [1].

For our test we use two single boards equipped with ARM SoC, the Firefly-RK3399 board[5] (one computational node of INCAS [2]), the ASUS Tinker board[6] and a general purpose x86 desktop. In order to fully exploit the ARM SoC (i.e. the workload across the whole CPU+GPU system), a direct N-body code, called ExaHiGPUs [3, 4, 5], is employed. Such a code is a re-engineered and optimized version of a widely used code [6], which is an implementation of the numerical integration of the classical, gravitational, N-body problem, based on a 6th order Hermite's integration scheme with block time steps, with a direct evaluation of the particle-particle forces.

The technical report is organized as follows. Section 2 describes the code used to perform the power measurements. In Section 3 we present the platforms used in our tests. Section 4 is devoted to present how the power measurements have been performed and to describe the experimental setup. Section 5 presents the results. Conclusions and future developments are exposed in Section 6.

2. Code profiling

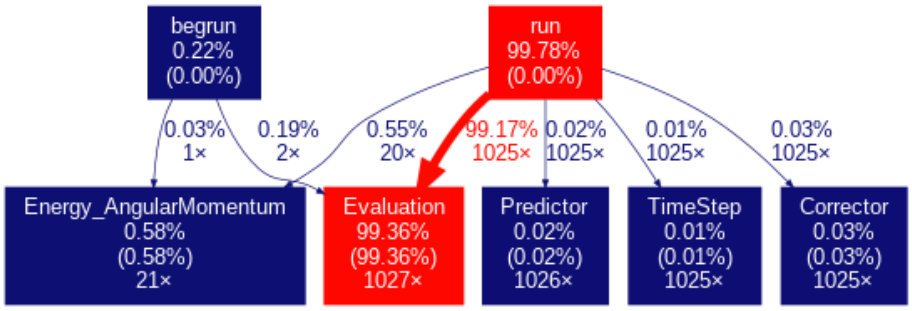

Figure 1: Call graph of ExaHiGPUs profiled using *gprof* tool. The figure shows nodes and edges above the threshold 0.01.

[5] http://en.t-firefly.com/Product/Rk3399/spec.html
[6] http://www.asus.com/Single-Board-Computer/Tinker-Board/specifications/

This work aims to compare the relative power efficiency between the CPU (single-core, dual-core, multi-core) and the GPU on SoC using a real scientific application (it is like to measure the energy footprint of a scientific application). The N-body application, called ExaHiGPUs, is able to exploit both CPUs and GPUs, and it is our candidate application to study the electric power consumption of such devices during a simulation run.

Actually, two version of the code are available (all details of the application in [3, 4, 5]):

i) `Standard C code:` cache-aware designed for CPUs and parallelized with hybrid MPI[7] + OpenMP[8] programming. The code has been instrumented to get the time spent executing the stages of the Hermite integrator;

ii) `OpenCL code:` conceived to target accelerators like both PCIe GPUs and GPUs on SoC. All the stages of the Hermite integrator are performed on the OpenCL-compliant device(s). The kernel implementation exploits *local* memory (OpenCL terminology) of device(s), which is generally accepted as the best method to reduce global memory latency in PCIe GPUs (hereafter we refer to this implementation as `PCIe-GPU-implementation`). However, on ARM GPUs on SoC, the `global` and `local` OpenCL address spaces are mapped to main host memory (as reported by the ARM developer guide[9]). So, a specific ARM-GPU-optimized version of all kernels of ExaHiGPUs, in which `local` memory is not used, has been implemented (hereafter we refer to this implementation as `SoC-GPU-implementation`). In the results shown in this technical report, we use the `PCIe-GPU-implementation` in order to exploit the NVIDIA-GPU and the `SoC-GPU-implementation` to exploit the Mali-GPU (devices are detailed in Section 3). Kernel execution times on the GPUs have been obtained by means of OpenCLs built-in profiling functionality, which allows the host to collect runtime information.

[7] https://www.open-mpi.org/
[8] https://www.openmp.org/
[9] `https://bit.ly/2T1yrrw`

Although ExaHiGPUs performs several calculations during a single time step, the 99% is spent on the *Evaluation* stage of the 6th order Hermite integration schema (serial application when I/O is disabled). That has been found profiling the whole application (`standard C` version of the code) by means of *GNU gprof* [7], the performance analysis tool for Unix applications. Then, the profiler output is visualized using *GProf2Dot*[10] tool as a colorful call graph that makes it easy to understand the statistics taking into account nodes and edges above a threshold value of 0.01, as shown in Figure 1. The profiling of the whole application is shown in the Appendix A. The code profiling results suggest to measure the energy consumption during the execution of the *Evaluation* kernel, in an infinite while loop[11].

2.1. Arithmetic precision considerations

Arithmetic precision plays a key role during the integration of the equations of motion of an N-body system. We have already demonstrated in a previous technical report [3] that both double-precision (hereafter DP) and emulated-double-precision (also called extended-precision, hereafter EX) arithmetic[12] are able to minimize the accumulation of the round-off error during the Hermite integration, so preserving the total energy of the system during the simulation.

3. Computing platforms

In this section we describe the three platforms used in our tests. In Table 1, we list the devices, and we highlight in bold the ones exploited in this technical report.

[10] https://github.com/jrfonseca/gprof2dot.

[11] Source code of the *Evaluation* kernel is shown in [5].

[12] An EX-number provides approximately 48 bits of mantissa at single-precision exponent ranges. More details in [3].

Platform	Firefly-RK3399 board	Asus Tinker board	x86 Desktop
SoC/Motherboard	Rockchip RK3399	Rockchip RK3288	ASUS P8B75-M LX
CPU	ARM 2x **A72** + 4x **A53** 64-bit	ARM 4x **A17** 32-bit	Intel 4x **i7-3770** 64-bit
GPU	ARM **Mali-T864**	ARM Mali-T764	NVIDIA **GeForce-GTX-1080**
RAM	4GB DDR3	2GB DDR3	16GB DDR3
OS	Ubuntu 16.04 LTS	Ubuntu 18.04.2 LTS	Ubuntu 18.04 LTS
Compiler	gcc version 7.3.0	gcc version 7.3.0	gcc version 7.3.0
OpenCL	OpenCL 1.2	OpenCL 1.2	OpenCL 1.2

Table 1: The main characteristics of the boards used in the test. The devices exploited are highlighted in bold.

3.1. Firefly-RK-3399 board

The Firefly-RK3399 single-board[13] is equipped with the big.LITTLE architecture: 4x(Cortex-A53) cores with 32kB L1 cache and 512kB L2 cache, and a cluster of 2x(Cortex-A72) high-performance cores with 32kB L1 cache and 1M L2 cache. Each cluster operates at independent frequencies, ranging from 200MHz up to 1.4GHz for the LITTLE and up to 1.8GHz for the big. The multi-processor-SoC (MPSoC) contains 4GB DDR3 - 1333MHz RAM. The MPSoC features also the OpenCL-compliant Mali-T864 embedded GPU that operates at 800 MHz.

3.2. Asus Tinker board

The Asus Tinker single-board[14] is equipped with the MPSoC Rockchip-RK3288 featuring a Quad core 1.8 GHz ARM Cortex-A17 32-bit and a 600 MHz Mali-T760 MP4 GPU. The MPSoC contains 2GB Dual Channel DDR3.

Even though the Mali-T760 MP4 GPU is an OpenCL-compliant device and it is able to perform the calculations required by the code, the profiling OpenCL-APIs fail to retrieve the 64-bit value that describes the current device time

[13] Available: https://en.t-firefly.com/Product/Rk3399.html

[14] https://www.asus.com/Single-Board-Computer/Tinker-Board/specifications

counter, raising an invalid OpenCL event. So, the Mali-T670 MP4 is not used in our tests.

3.3. x86 Desktop

The commodity x86 desktop is equipped with a Quad core Intel core i7-3770 64-bit running at 3.9 GHz with 32kB L1, 256kB L2, and 8192 kB L3, and an PCIe NVIDIA GeForce-GTX-1080 GPU with 8 GB GDDR5X dedicated RAM, and 2560 CUDA cores running at 1733 MHz. The ASUS P8B75-M LX motherboard hosts 16GB of DDR3.

4. Measuring the Energy Consumption

We took the following steps in order to minimize the inaccuracies in estimating the current consumed by the CPU and the GPU while running the kernel. In Appendix B we clarify the units of measurements.

- Single-boards:
  - to avoid dynamic device frequency scaling and to maximize performance, we set the frequency governor to performance level;
  - to avoid the power draw by the AC-to-DC transformer, which makes the readings more noisy and spread out, the boards are powered by a DC power supply (Keysight E3634A);
  - after booting up the platform, we measure its stable current while the system is in idle. This gives us the $I_{baseline}$ consumption by the system;
  - $I^{device}_{baseline}$ is the total current consumed by the system running a given code implementation using a particular device (CPU or GPU), including the idle energy;
  - $I^{device} = I^{device}_{baseline} - I_{baseline}$ is the current that we are interested in;
  - $I^{device}_{baseline}$ and $I_{baseline}$ are the mean values over a range of three minutes.

- $E^{device} = V \times I^{device} \times T^{device}$ is the energy consumed by a given implementation of the kernel (energy-to-solution), where V and T^{device} are the voltage and the kernel running time (time-to-solution averaged over ten runs), respectively (voltage is constant, namely $V = 12$ Volt).

- Desktop:
  - we set the frequency governor to performance level;
  - the electric power draw is measured by means of a power meter (Yokogawa WT310E);
  - after booting up the platform, we measure the energy consumed in idle during a period of three minutes, giving us the $W_{baseline}$ of the system;
  - $W^{device}_{baseline}$ is the electric power drawn by the system running a given code implementation using a particular device (CPU or GPU) over a period of three minutes (ΔT_3);
  - the power drawn by the dedicated GPU (Nvidia GeForce-GTX-1080) is also monitored by a current probe (Fluke i30s);
  - $W^{device} = (W^{device}_{baseline} - W_{baseline}) \times T^{device} / \Delta T_3$ are the watts hours (energy-to-solution) that we are interested in, where T^{device} is the kernel running time (time-to-solution averaged over ten runs).

4.1. Experimental setup

To measure the current consumption of the devices under test, two simple setups were used, depending on the power supply type of the device.

- Devices powered by Direct current:
  - Benchtop Laboratory Power Supply, Keysight model E3634A;
  - Benchtop Multimeter, Hewlett Packard model 34401A;
  - AC/DC Current clamp, Fluke model i30s;

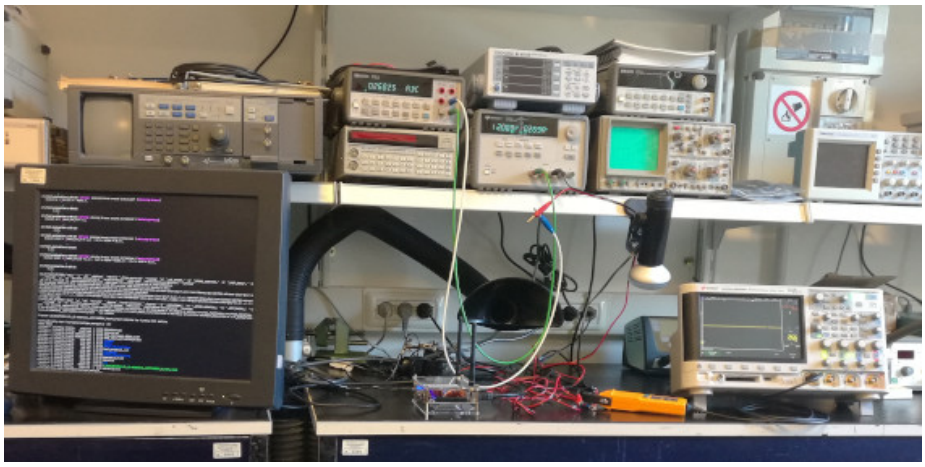

Figure 2: Experimental setup in the laboratory at OATS Basovizza Observing Station.

  - Digital Storage Oscilloscope, Keysight model MSOX3024T.

The benchtop laboratory power supply was set at the nominal supply voltage for the system and the multimeter was connected in series to measure the current flow. The output used in our test is the mean value of 300 measurements taken at each run with a sample rate of 5 Hz (1 sample every 200 msec). The oscilloscope was used to measure the dynamic behaviour of the current consumption taken by means of the current clamp. These devices were used just to monitor that the measurements are taken under almost constant load. Figure 2 shows the experimental setup, while the circuit used is shown in Figure 3. As final remark, we have also measured the efficiency of the AC power supply of the Firefly-RK3399 board provided by the vendor. The AC power supply efficiency is $\simeq 85\%$ in idle, while is $\simeq 91\%$ at full workload.

- Devices powered by Alternate current (mains supply):
  - Digital Power Meter, Yokogawa model WT310E;
  - AC/DC Current clamp, Fluke model i30s;
  - Digital Storage Oscilloscope, Keysight model MSOX3024T.

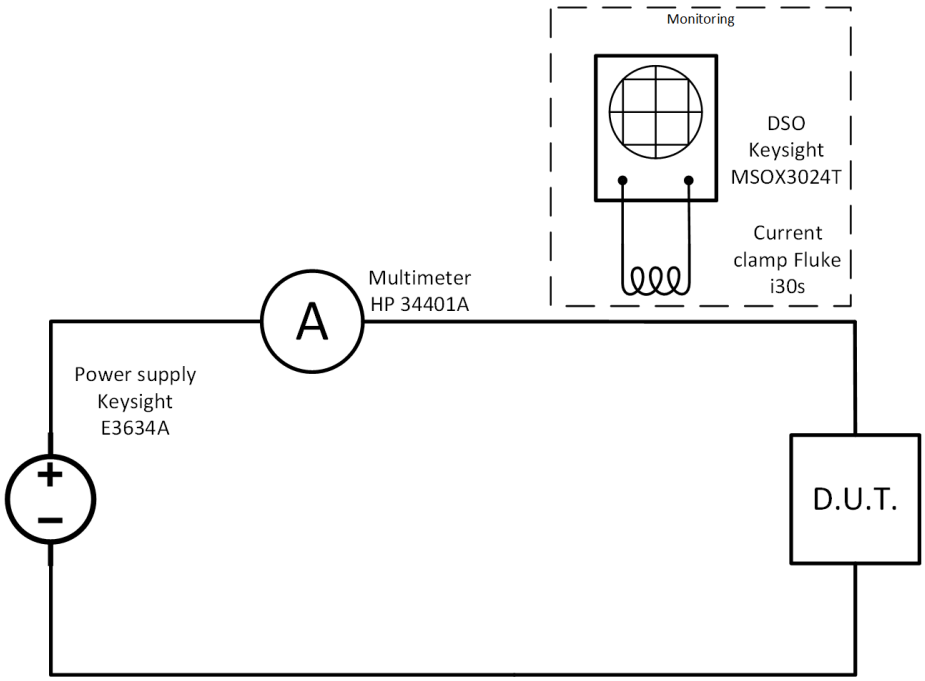

Figure 3: Direct current measurement circuit for the device under test (D.U.T.).

In this case, the systems were powered by their own power supply and the measurements were taken at the 230V mains input. The Power meter integrates the total power used during the chosen time period, and the energy-to-solutions (both W^{device} and $W_{baseline}$) rely on these measurements obtained by means of the power meter. For completeness of our tests, the dynamic behaviour of the current consumption of the PCIe GPU was also monitored using an oscilloscope and a current clamp measuring the auxiliary power supply input of the GPU. Since the GPU is powered by both the PCIe connector and the auxiliary power supply, the measurements taken with the current clamp intercept only a part of the total energy required by the GPU for the computation (i.e. without the power provided by the PCIe connector), that we quantify in $\simeq 70\%$ of the total. The measurement circuit is shown in Figure 4.

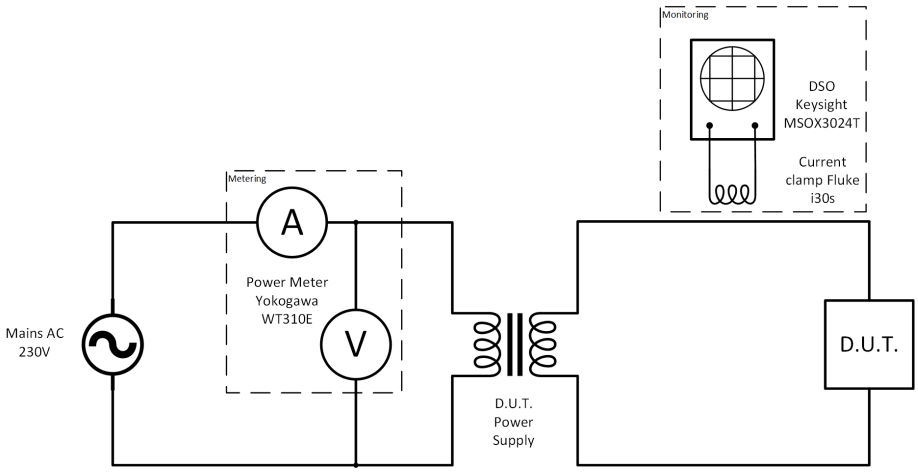

Figure 4: Alternate current measurement circuit for the device under test (D.U.T.).

5. Results

Energy-to-solutions and time-to-solutions are obtained running the application for one time step using 65536 particles and both DP and EX arithmetic (details in Section 2.1).

Figure 5 shows measurements for DP arithmetic. The most effective device, in terms of time-to-solution, is the PCIe NVIDIA-GeForce-GTX-1080 GPU.

Regarding CPUs, the time-to-solution scales linearly with the number of cores exploited, and saturates when the number of OpenMP threads exceeds the available cores, as expected. Multi-core implementation is always the most effective solution, both in terms of time-to-solution and energy-to-solution. It is worth noting that dual-core ARM-Cortex A72, running at 1.80 GHz, is $\simeq 4.4$ times more power-efficient than the single-core Intel-i7, running at 3.40 GHz, with a time-to-solution increased by 17% (we are comparing in Figure 5 the red circle for the ARM-A72 with the violet triangle for the Intel-i7).

Figure 6 shows measurements for EX arithmetic. In this case, both GPUs outperform CPUs.

To better study the effect of the EX arithmetic, in Figure 7 we show the ratio of time-to-solution between DP and EX arithmetic. The results are shown

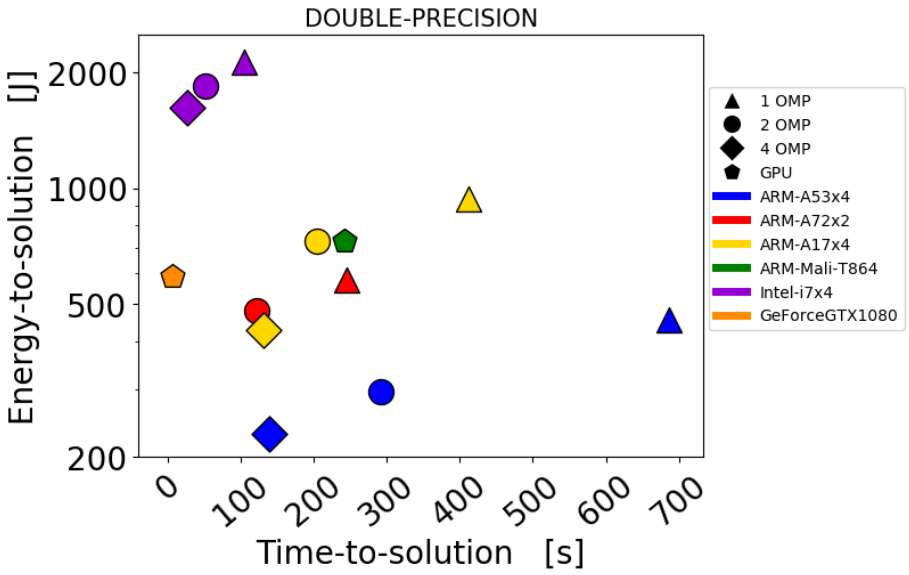

Figure 5: Energy-to-solution (in Joule) as a function of time-to-solution (in second) for DP arithmetic. Blue symbols for ARM-A53x4 CPU, red symbols for ARM-A72x2 CPU, gold symbols for ARM-A17x4 CPU, green symbols for ARM Mali-T864 GPU on SoC, violet symbols for Intel-i7x4 CPU and orange symbols for GeForceGTX1080 PCIe GPU. Triangle up for 1 OMP thread (serial calculation), circle for 2 OMP threads, diamond for 4 OMP threads, and pentagon for GPU kernel implementation (work-group size of 64).

exploiting all the available CPU cores. The performance improvement is a factor of $\simeq 2$ for the Mali-T864 GPU on SoC and $\simeq 20$ for the GTX-1080 PCIe GPU, while all CPUs suffer a significant performance degradation.

The results show that, basically, PCIe GPUs for gaming, like the NVIDIA-GTX-1080, and GPUs on SoC are suitable for 32-bit arithmetic. DP arithmetic is resource-eager on such devices and the EX arithmetic can be a trade-off in order to efficiently exploit such GPUs, actually not designed for HPC.

6. Conclusions and future developments

The energy footprint of scientific applications will become one of the main concerns in the HPC sector. SoC technology is specifically designed to optimize, among others, the energy-to-performance ratio. In this first work, we have

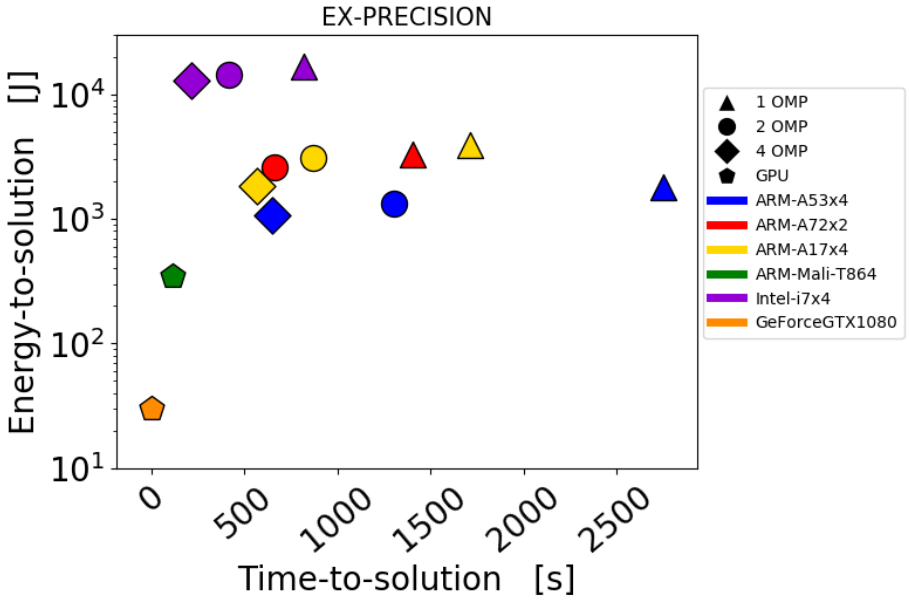

Figure 6: Energy-to-solution (in Joule) as a function of time-to-solution (in second) for EX arithmetic. Blue symbols for ARM-A53x4 CPU, red symbols for ARM-A72x2 CPU, gold symbols for ARM-A17x4 CPU, green symbols for ARM Mali-T864 embedded GPU, violet symbols for Intel-i7x4 CPU and orange symbols for GeForceGTX1080 dedicated GPU. Triangle up for 1 OMP thread (serial calculation), circle for 2 OMP threads, diamond for 4 OMP threads, and pentagon for GPU kernel implementation (work-group size of 64).

started to explore the impact of software design of a scientific application on the energy-to-solution and time-to-solution exploiting low-cost SoC-based platforms. We have shown that SoC technology is emerging as a valid alternative to "traditional" technology for HPC, which is focused more on peak-performance than on power-efficiency. However, code parallelization and optimizations are mandatory in order to fully exploit heterogeneous SoC platforms, but they are not so widely applied by the scientific community.

In future works, we will assess the energy footprint of other aspects of the application, such as the network and the I/O.

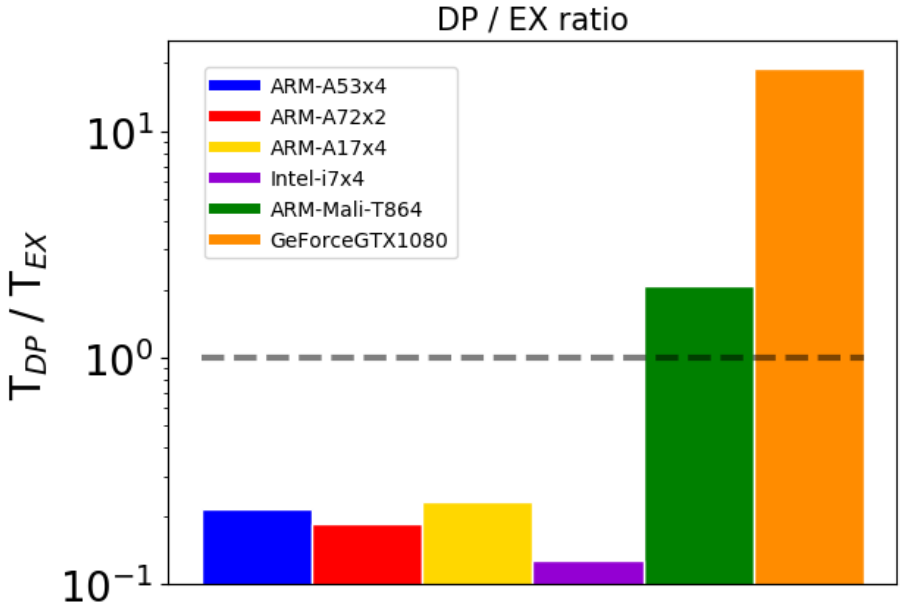

Figure 7: The ratio of the time-to-solution between DP arithmetic and EX arithmetic for all devices. The results are shown exploiting all the available CPU cores.

7. Acknowledgments

The technical report was carried out within the EuroEXA and ExaNeSt FET-HPC projects (grant no. 754337 and no. 671553), and the ASTERICS project (grant no. 653477), funded by the European Unions Horizon 2020 research and innovation programme.

This work has been made use of IPython [8], Numpy [9] and MatPlotLib [10].

References

[1] G. Taffoni, G. Murante, L. Tornatore, D. Goz, S. Borgani, M. Katevenis, N. Chrysos, M. Marazakis, Shall numerical astrophysics step into the era of Exascale computing?, in: Astronomical Data Analysis Software an Systems XXVI (ADASS XXVI), Astronomical Society of the Pacific Conference Series, 2016.

[2] S. Bertocco, D. Goz, L. Tornatore, G. Taffoni, INCAS: INtensive Clustered ARM SoC - Cluster Deployment, in: INAF-OATs technical report, 222, August 2018. doi:10.20371/INAF/PUB/2018_00004.

[3] D. Goz, L. Tornatore, S. Bertocco, G. Taffoni, Direct N-body code designed for heterogeneous platforms, in: INAF-OATs technical report, 223, July 2018. doi:10.20371/INAF/PUB/2018_00002.

[4] D. Goz, L. Tornatore, S. Bertocco, G. Taffoni, Direct N-body code designed for a cluster based on heterogeneous computational nodes, in: INAF-OATs technical report, 224, August 2018. doi:10.20371/INAF/PUB/2018_00005.

[5] D. Goz, S. Bertocco, L. Tornatore, G. Taffoni, Performance of direct N-body code on ARM64 SoC, in: INAF-OATs technical report, 241, December 2018. doi:10.20371/INAF/PUB/2018_00006.

[6] M. Spera, R. Capuzzo Dolcetta, D. Punzo, HiGPUs: Hermite's N-body integrator running on Graphic Processing Units (Jul. 2012). arXiv:1207.002.

[7] S. L. Graham, P. B. Kessler, M. K. Mckusick, Gprof: A call graph execution profiler, SIGPLAN Not. 17 (6) (1982) 120–126. doi:10.1145/872726.806987.
URL http://doi.acm.org/10.1145/872726.806987

[8] F. Perez, B. Granger, Ipython: A system for interactive scientific computing, Computing in Science Engineering 9 (3) (2007) 21–29. doi:10.1109/MCSE.2007.53.

[9] S. van der Walt, S. Colbert, G. Varoquaux, The numpy array: A structure for efficient numerical computation, Computing in Science Engineering 13 (2) (2011) 22–30. doi:10.1109/MCSE.2011.37.

[10] J. Hunter, Matplotlib: A 2d graphics environment, Computing in Science Engineering 9 (3) (2007) 90–95. doi:10.1109/MCSE.2007.55.

Appendix A. ExaHiGPUs profiling

In Figure A.8 the profiling of the entire application obtained by means of *GNU gprof*[15] and *GProf2Dot*[16] tools is shown.

Appendix B. Units of measurement

In this technical report the energy-to-solution refers to the total energy required to perform a given calculation.

The joule is a derived unit of energy in the International System of Units (SI). In terms of SI units, a joule is defined as below:

$$1\,\texttt{Joule} = 1\,\texttt{Ampere} \cdot 1\,\texttt{Volt} \cdot 1\,\texttt{second} \tag{B.1}$$

In the case of single-board platforms, we measure the electrical current (in Ampere), and the computing time (in seconds), while the voltage is constant. The energy-to-solution is obtained applying the Equation B.1.

In the case of the desktop, we measure the energy W consumed over a period of three minutes (in W*h) and the computing time T (in seconds). Given that, the energy-to-solution E (in Joules) is obtained applying the following equation:

$$E = (W * 3600) * (T/180) \tag{B.2}$$

[15] https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_mono/gprof.html
[16] https://github.com/jrfonseca/gprof2dot.

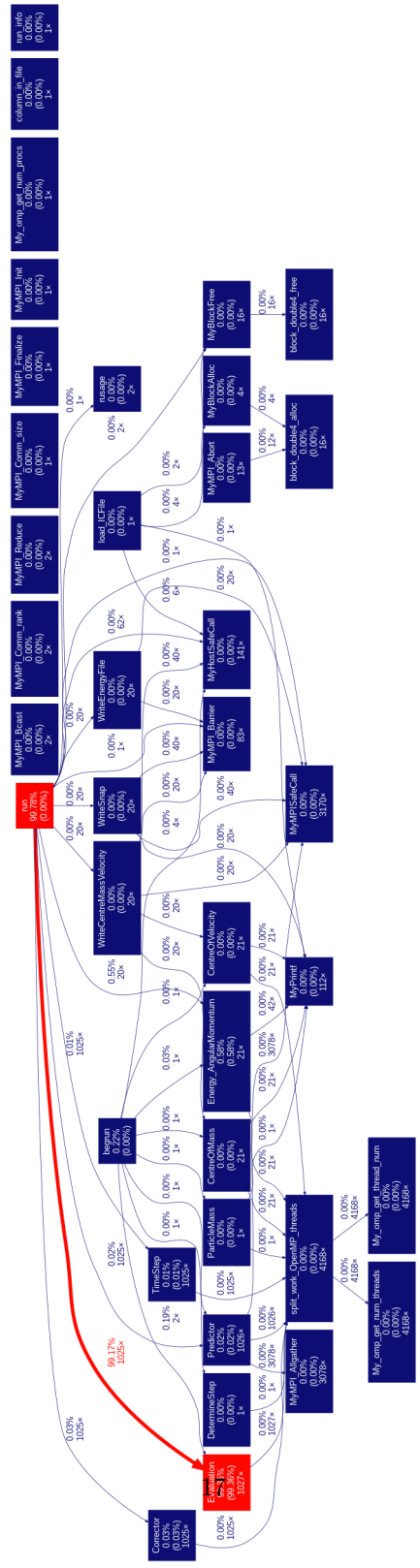

Figure A.8: Call graph of ExaHiGPUs profiled using *GNU gprof* and *GProf2Dot* tools.