Low power high performance computing on Arm system-on-chip in Astrophysics

Giuliano Taffoni[1], Sara Bertocco[1], Igor Coretti[1], David Goz[1], Antonio Ragagnin[1], and Luca Tornatore[1]

National Institute of Astrophysics - Astronomical Observatory of Trieste, via G.B. Tiepolo 11, Trieste, Italy,
giuliano.taffoni@inaf.it,

Abstract. In this paper we quantitatively evaluate the impact of computation on the energy consumption on Arm MPSoC platforms, exploiting both CPUs and embedded GPUs. Performance and energy measures are made on a a direct N-body code, a real scientific application from the astrophysical domain. The time-to-solutions, energy-to-solutions and energy delay product using different software configurations are compared with those obtained on a general purpose x86 desktop and PCIe GPGPU. With this work we investigate the possibility of using commodity single boards based on Arm MPSoC as an HPC computational resource for real Astrophysical production runs. Our results show to which extent those boards can be used and which modification are necessary to a production code to profit of them. A crucial finding of this work is the effect of the emulated double precision on the GPU performances that allow to use embedded and gaming GPUs as excellent HPC resources.

Keywords: Arm, GPU, MPSoC, HPC, energy-to-solution, Energy Delay Product

1 Introduction

In the last decade, energy efficiency has become a main concern in the High Performance Computing (HPC) sector. These systems are built using power hungry high performance systems, and their high energy consumption poses major hedges for achieving exascale computation. Energy efficiency is both a fundamental requirement of large scale platforms and one of the main challenges for future processors, interconnect and storage design [17]. In fact, the eligibility of a exascale computing system must not only pass through performance assessment of its hardware but also of its energy usage.

Commodity single board computers are an interesting case of heterogeneous systems to be utilized for energy efficiency studies. These are low cost single circuit board computers that embed CPUs, GPUs, memory, storage, general purpose I/O ports for external devices and expansions (e.g. SD card connects, USB, PCIe, HDMA, etc). The HPC community is already studing the use of those low-powered System-on-Chip (SoC) architectures in large-scale HPC systems,

trying to reach production-ready solutions. Additionally, various companies are also studying one-board computers equipped with different hardware solutions and based on Multi-processing System-on-Chip (MPSoC).

This work was done in the framework of ExaNeSt and EuroExa European funded project aiming at the design and development of a prototype of an exascale HPC facility based on low power Arm SoC and FPGAs as accelerators [13, 14].

Although the performance of these machines has been profiled in the context of benchmarking tools [19], in this work we study the performance on a real code, an N-Body solver for astrophysical simulations. Our goal is to investigate the trade-off between time-to-solution and energy-to-solution when using real full production runs, and the problems that a developer can face when approaching this kind of platforms. In doing this study, we analyze the computing capabilities and the relative power efficiency between the CPU (single-core, dual-core, multi-core) and the GPU on a MPSoC produced by Rockchip's Firefly-RK3399. We further compare these results with a "standard architecture" based on an Intel server with a GPGPU.

To our knowledge, this paper provides the first comprehensive evaluation of a real astrophysics application on single board computers and in particular on the MPSoC Rockchip Firefly RK3399.

The paper is organized as follows. In section 2 we introduce previous results in the literature that shaped this work. In Section 3 we describe the code and we discuss strategies adopted in order to port and optimize a state-of-the-art N-body code on heterogeneous platforms. In Section 4, we present the single board MPSoC computers that we have identified for our tests and we discuss our choice to use the Firefly-RK3399 board. In Section 5 we discuss the methodology we used to make the performance and energy tests, including some considerations on our choices of architectures for HPC and the role of double precision arithmetic. In Section 6 we discuss the performance measurements for all the platforms. Section 7 is dedicated to the power consumption analysis including a description of the experimental setup. Energy measurements results are discussed in Section 8. Last Sections are dedicated to the conclusions.

2 Related works

SoC devices are experiencing a growing interest because of their versatility, low-power consumption and their low cost. This is showed, for instance, by the success of one of the first single board computers: the Raspberry Pi [31]. Today there exists a large number of alternatives to Raspberry Pi, and various companies are investing on boards that are equipped with different hardware solutions and are based on MPSoC architectures (in Section 4 we will present some of them).

MPSoC integrated circuits are composed of asymmetric multi-core systems combined with graphic-processing units (GPUs) aiming to optimize the energy-to-performance ratio. MPSoC are mainly deployed for the mobile market, al-

though they have been recently utilized in sectors where traditional resources would not be appropriate or in situations where a standard computing platform would not be suitable: for instance, educational purposes [30, 8], HPC or Cloud [25, 9, 1], expendable computers [32], sensors networks [15, 20] and Fog computing [5].

MPSoC devices are particularly interesting as they implement heterogeneous architectures where multi-core CPUs and GPUs are coupled with a unified memory system (UMS) where expensive copy operations that exchanges data between the host and the device are not required.

In the last years, GPUs became widely used in scientific programming in order to accelerate computational demanding applications with extensive data-level parallelism because they offer high floating-point throughput and memory bandwidth. In the past, they have had limited device memory, until recently, when their on-board capacity has grown up to several GBs. Though, in general, the capacity of a GPU memory is significantly lower than its host memory. For this reason programmers are obliged to work with two memory spaces and move data from one to the other memory space with an impact in performances and energy consumption. This is crucial for applications striving to solve larger and larger problems. There exists already some solutions to solve this limitation (e.g. POWER8 with NVLink CPUs with four NVIDIA[1]), however the UMS of MPSoC boards may represent an interesting low power and low price solution.

Some authors already analyze MPSoC performance and Energy consumption using standard benchmarks (e.g. HPL, HPCG, DGEMM) [24]. Instead of benchmarking our target platforms using standard suites tuned to measure peak performances. Instead, we are probing the platforms by using real scientific applications and real production runs. In fact, it is well known that, in some case, codes are able to use only a few percentage of the peak Floating point operations/second (FLOPs) [16], in particular memory bound codes as numerical cosmological simulations or data reduction analysis programs.

3 *N*-body Astrophysical codes

In astrophysics, the N-body problem is the problem of predicting the individual motions in a group of celestial objects interacting with each other gravitationally. This applies mainly to the study of the dynamic of star clusters and globular clusters [27].

The numerical solution of the direct N-body problem is still considered a challenge despite the significant advances in both hardware technologies and software development. The main drawback related to the direct N-body problem relies on the fact that the algorithm requires $O(N^2)$ computational cost. There are some N-body codes designed for real scientific production in astrophysics using CPUs or GPUs [4, 12, 21, 18, 7, 26]. None of the above has been ported or optimized for embedded GPUs.

[1] https://www.ibm.com/blogs/systems/ibm-nvidia-present-nvlink-server-youve-waiting/

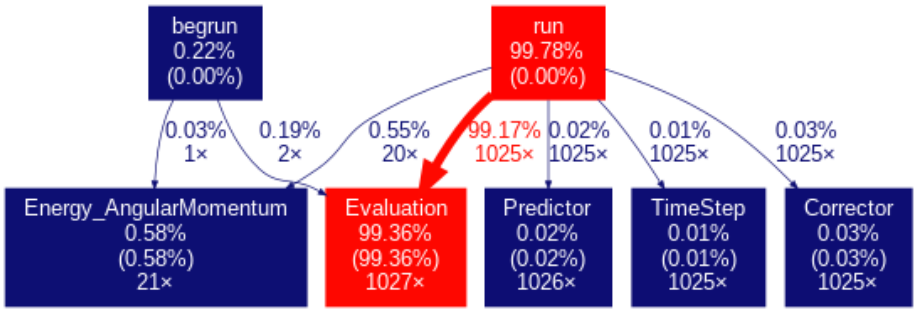

Fig. 1: Call graph of Hy-Nbody profiled using *gprof* tool. The figure shows nodes and edges above the threshold 0.01.

3.1 The Hy-Nbody code

Our Hy-Nbody code [11] is a modified version of a GPU N-body algorithm [7, 26], based on high order Hermite integration schema [22] using a block time-stepping.

In our implementation, the GPU is fed by the host CPU with the gravity equation of data in the form of coordinates, velocities and masses of particles, and it handles calculating the forces for the data points. Differently from other N-Body codes, we design the algorithm to fully exploit the compute capabilities of heterogeneous architectures. The Hermite schema is implemented and optimized using OpenCL kernels, allowing to test the code on any OpenCL-compliant device (e.g. CPUs and GPUs). We use a fine grained parallelization approach: the host code is parallelized with hybrid MPI+OpenMP programming, while the device code is parallelized with OpenCL. The user is allowed to choose at compile time if the application uses MPI or OpenMP, or both, or neither. The Hermite integration is performed on the selected OpenCL-compliant device(s) and all kernels of the application have been vectorized improving memory bandwidth and reducing the number of loads/stores.

3.2 Profiling Hy-Nbody

Code profiling reveals that 99% of the time is spent on the *Evaluation* stage of the 6th order Hermite integration schema (serial application when I/O is disabled), as shown in Figure 1.

The floating point operations (FLOPs) of the Hy-Nbody code have been evaluated through the Performance Application Programming Interface (PAPI) tool [28], allowing us to estimate the arithmetic intensity (ratio of FLOPs to the memory traffic) of the *Evaluation* kernel as $I \simeq 1.5 \cdot 10^7/N$ [FLOPs/byte], with N the number of particles.

Following the *Roofline Model*[2], each kernel is going to be either memory-bound or compute-bound on a specific architecture, since performance is upper bounded by both the peak flop rate, and the product of streaming bandwidth and the flop to byte ratio. The peak performance of a platform can be usually derived from architectural manuals, while the peak bandwidth, which references to peak DRAM bandwidth to be specific, is instead obtained via benchmarking. However, both code profiling and arithmetic intensity estimate suggest that the *Evaluation* kernel is compute-bound on every architecture with $N \sim 10^4 - 10^5$, which is the typical number of particles assigned to a device during a production run.

4 A single board computer for HPC: Firefly-RK3399

To fully exploit the MPSoC heterogeneous boards for scientific calculations, it is necessary to use hardware solutions that offers at least: (i) double-precision floating point arithmetic, (ii) options for high performance I/O and memory interface, (iii) full support for a parallel programming model as CUDA [23], OpenCL [3] or OpenACC [10].

The latest Arm MPSoC boards satisfy these requirements as they support 64-bit floating-point arithmetic precision operations and OpenCL 1.2 specifications.

Those boards are based on the so called Arm big.LITTLE architecture. It features two sets of cores: a low performance energy-efficient cluster (the LITTLE one), and a power-hungry high-performance cluster (the big one). Big and LITTLE cores support the same instruction set, so they can run the same binaries and therefore are easily combined within the same system. Even if extremely promising from the energy efficiency point of view, this kind of heterogeneous boards are extremely complex to exploit for scientific applications. They require to design codes that are able to (i) optimize the scheduling of big.LITTLE cluster, (ii) the use of GPUs, (iii) the memory access.

Nowadays, there is a wide variety of single board computers available on the market each with its own characteristics. We analyzed some of them and identified the most promising board for our analysis. On the basis of a set of requirements, our candidate should accomplish:

- at least 4GB of RAM: our software is quite demanding in terms of RAM in particular as we are willing to test some scientific full production runs;
- OpenCL capable GPU device as our code has been re-design in OpenCL to exploit heterogeneous platforms;
- commodity hardware with Linux OS support;
- ease of expansion. A single board computer that can be expanded with PCI or USB devices (e.g. external disks or network cards);
- on-board gigabit Ethernet. This will allow future expansion towards a cluster of boards.

[2] Roofline is a visually intuitive performance model used to bound the performance of various numerical methods and operations running on multicore, manycore, or accelerator processor architectures.

Table 1: A comparison of the Single Board platforms based on the MPSoC that we identifed for our tests.

Board	**SoC**	**Arch**	**RAM**	**GPU**	**NET**
Raspberry Pi 3 B+	Broadcom BCM2837B0, Cortex-A53 @ 4x1.4GHz	64-bit	1GB	VideoCore IV	1GB over USB
Odroid XU4	Exynos 5422 Cortex-A15 and Cortex-A7 42.1GHz&41.5GHz	32-bit	2GB	Mali-T628 MP6	1GB
Banana Pi M64	Allwinner A64 Cortex-A53 @ 4x1.4GHz	64-bit	2GB	Mali-400 MP2	1GB
Pine A64	Allwinner r18 Cortex-A53 @ 4x1.3GHz	64-bit	2GB	Mali-400 MP2	1GB
Asus Tinker Board	Rockchip RK3288 Cortex-A17 @ 4x1.8 GHz	32-bit	2GB	Mali-T760 MP4 OpenCL 1.1	1GB
Firefly RK3399	Rockchip RK33399 Cortex-A53 and Cortex-A72 41.4GHz&22.0GHz	64-bit	2/4GB	Mali-T864 MP4 OpenCL 1.2	1GB

All the boards listed in Table 1 are based on Arm SoC but only a few of them implement a big.LITTLE architecture and only one has enough memory to satisfy our requirements: the Firefly-RK3399 board.

The Firefly-RK3399 single board computer has 6 core 64-bit Arm big.LITTLE SoC architecture. The board contains a cluster of four Cortex-A53 cores with 32kB L1 cache and 512kB L2 cache, and a cluster of two 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 SoC contains 2 or 4GB DDR3 - 1333MHz RAM. The L2 caches are connected to the main memory via the 64-bit Cache Coherent Interconnect (CCI) 500 that provides full cache coherency between big.LITTLE processor clusters and provides I/O coherency for the Mali-T864 GPU. The peculiarity of this board is that Mali-T864 is a OpenCL-compliant Quad-Core Arm Mali GPU.

This board is an Open Source platform with excellent expansion capabilities, it is equipped with 4 USB2.01 USB3.01 USB3.0 Type-C, a MicroSD (TF) Card Slot and an HDMI video connector. The network card is a Realtek RTL8211E 10/100/1000 RJ-45 interface and it also has a PCIe Next Generation Form Factor M.2 connector.

There are different RAM and storage sizes available, and we decide to test the 4GB of RAM and 16GB of High-Speed eMMC configuration.

The Board is installed with Ubuntu 16.04 LTS Linux distribution and OpenCL 1.2.

The host hardware we used to develop and validate the application and to compare with the MPSoC device, is a workstation with one Intel Core i7-3770x4 running at 3.40 GHz and one Nvidia GeForce-GTX-1080 graphics card in the

Table 2: The main characteristics of the board used in the test and of the Desktop.

Platform	Firefly-RK3399 Board	Desktop
	Rockchip RK3399	ASUS P8B75-M LX
`CPU`	Arm A72x2 + A53x4 64-bit	Intel i7-3770x4 64-bit
`GPU`	Arm Mali-T864	NVIDIA GeForce-GTX-1080
`RAM`	4GB DDR3	16GB DDR3
`OS`	Ubuntu 16.04 LTS	Ubuntu 18.04 LTS
`Compiler`	gcc version 7.3.0	gcc version 7.3.0
`OpenCL`	OpenCL 1.2	OpenCL 1.2

PCI Express (16) bus. The workstation runs a Linux Ubuntu SMP kernel version 4.15.0-20 generic and graphics card driver NVIDIA 390.48. In Table 2 we describe the main characteristic of the MPSoC platform used in our tests.

5 Considerations and methodology

The big.LITTLE architecture has been conceived for mobile devices where applications are developed and optimized in order to stick to low energy consumption for routinely workloads and to benefit to more performant cores on demand. At odds, a scientific code is designed to work on homogeneous core sets and to steadily extract maximum performance from them. Our application is in fact conceived to run on homogeneous platforms and typically operate by dividing the workload on even units. Executing these equal work units on an asymmetric system is expected to degrade the overall performance due to load imbalance.

In our case, we limit the load imbalance binding the OpenMP threads to a defined CPU cluster, setting explicit core affinity. We use Linux `taskset` command to choose the affinity. This way, we avoid performance degradation due to the thread migration from a cluster to the other and we measure the performances and energy of each one of the two CPU clusters homogeneously. Anyway our code is designed to maximize the performance, so we cannot benefit of the cluster migration features for energy saving.

While ARM cores does not have frequency throttling ability, in our experiments we freeze the frequencies at a given level using the `performance` scaling governor, in order to prevent the dynamical scaling of the cores frequencies during runtime. Furthermore, to stabilize the results and to average possible system fluctuation, each run of the code is repeated 10 times and the results presented later in the paper are always the averaged values. Errors are reported explicitly when greater than 1%.

On the MPSoC architectures GPUs, the global and local OpenCL address spaces are mapped to main host memory. This means that explicit data copies from global to local memory and associated barrier synchronizations are not

necessary. Thus, using local memories as a cache can waste both performance and power. For this reason, specific Arm-optimized version of all kernels of HY-NBODY has been implemented in which the local memory is not used.

5.1 Floating point arithmetic considerations

The Hermite 6th order integration schema requires double precision (DP) arithmetic in the evaluation of inter-particles distance and acceleration in order to minimize the round-off error. Full IEEE-compliant DP-arithmetic is efficient in market available CPUs, but it is still extremely resource-eager and performance-poor in other accelerators like gaming or embedded GPUs. The theoretical best case for DP performance is 1:2 SP, simply because it involves computing with double the number of bits as FP32. However, this ratio can me much lower for many devices[3].

As an alternative, the extended-precision (EX) (or emulated double precision) numeric type [29] can represent a trade-off in porting HY-NBODY on devices not specifically designed for scientific calculations. An EX-number provides approximately 48 bits of mantissa at single-precision exponent ranges. HY-NBODY can be compiled using DP, EX or single precision (SP) arithmetic (user-defined at compile time).

The energy E and the angular momentum L of the N-body system during the simulation are constant quantities and we use them to evaluate the effect of the arithmetic on the accumulation of the round-off error. The stability of the computation is verified with an error lower than 10^{-8} for DP and EX, while SP calculations do not conserve neither E nor L, and therefore we do not consider them suitable for real scientific full production runs. Since in this work we are interested in benchmarking real scientific cases, we do not include any SP result.

6 Computational performances: CPUs and GPUs

First we measure and investigate the CPUs speedup, i.e. the ratio of the serial execution time to the parallel execution time utilizing multiple cores by means of OpenMP threads. We run our code varying the number of particles in the N-body integration for different OpenMP threads, pinning the processes first to the 4 cores of the Arm Cortex-A53 and then to the 2 cores of the Cortex-A72.

Figure 2 shows the speedup for both Arm Cortex-A53x4 and Cortex-A72x2 CPUs varying the number of OpenMP threads as a function of the number of particles. As expected, the best performance is achieved when each core handles one thread. Time-to-solution saturates when the number of OpenMP threads exceeds the available cores. In the case of Arm Cortex-A53x4, we observe super-linear scalability that may be due to an optimized use of the caches.

As for the measure of the Mali GPU performance, we first note that, as already mentioned, we never use the local memory to avoid the cost of memory

[3] https://www.geeks3d.com/20140305/amd-radeon-and-nvidia-geforce-fp32-fp64-gflops-table-computing/

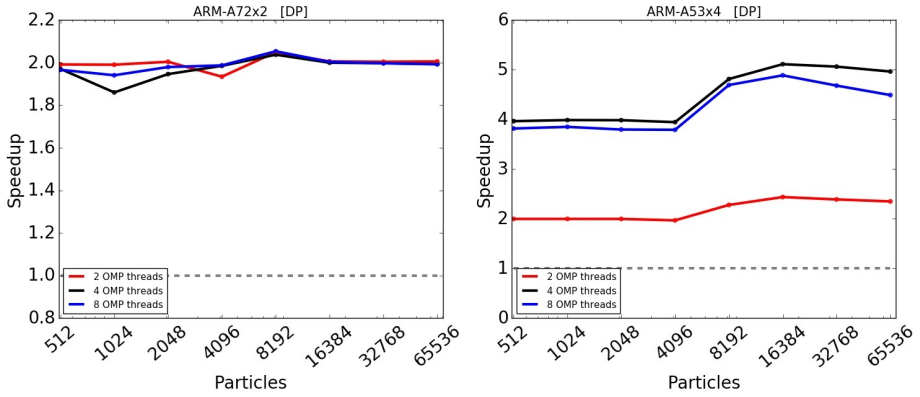

Fig. 2: Host speedup for DP-arithmetic as a function of the number of particles. We vary the number of OpenMP threads. Left panels for Arm Cortex-A72x2 and right panels for Arm Cortex-A53x4.

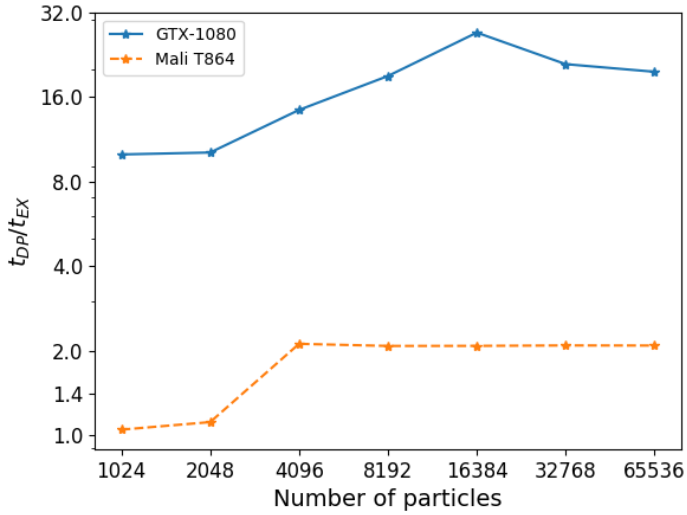

Fig. 3: The ratio between the execution time of EX arithmetic and DP arithmetic as a function of the number of particles for both Mali-T864 and Nvidia GeForce-GTX-1080 GPUs.

copy. Following the Arm OpenCL Developer Guide [2], we test the effects of vectorizing the code increasing the size of work-group. We run Hy-Nbody varying the work-group size from 4 up to 256 and we measure the execution time increasing the number of particles from 1024 up to 65536.

Kernel execution times on the GPU have been obtained by means of OpenCLs built-in profiling functionality, which allows the host to collect runtime information. Despite Arm recommends for best performance using a work-group size that is between 4 and 64 inclusive, our measurement shows that the execution time is not driven by any specific work-group size (the execution time is constant within 3%). The work-group size is not even affecting the energy consumption so, from now on, we fix the work-group size at 64.

Finally, we measure the DP and EX performance of both Mali-T864 and GeForce-GTX-1080 and compare it with the SP performance executing a set of runs varying the number of particles from 1024 to 65536. The DP/FP ratio for the GTX-1080 is $\sim 1/32$ as also discussed by Geeks3D blog, while for the Mali GPU it is $\sim 1/10$ (see Table 3 for more details).

On the other hand the effect on performance of the EX arithmetic is extremely important. In Figure 3, we present the result of a set of simulations increasing the number of particles. The effect of the EX becomes evident from 2048 particles and it stabilizes from 4096 particles on. The performance improvement is ~ 2 for the Mali GPU and ~ 20 for GTX-1080. For the GTX-1080, tests has been done using a work-group size of 64, however, differently from Mali GPU, the size of the work-group significantly affects the execution time; with EX arithmetic the best performances are obtained for work-group size of 64 and 128.

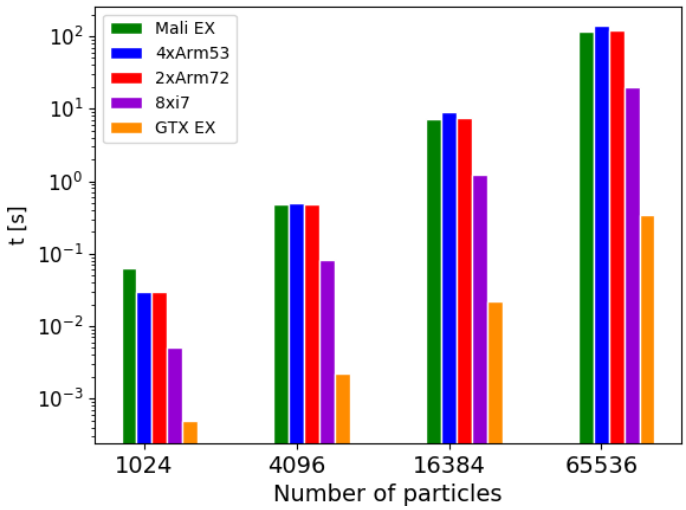

Fig. 4: The execution time in seconds for different devices as a function of the number of particles.

In Figure 4, we compare the time-to-solution in seconds for different devices. For each device we plot the configuration that optimizes the performances in terms number of cores, work-group size, DP for CPUs and EX for GPUs. A summary of the configuration and results is presented in Table 3. We note that, from a pure performance point of view the Nvidia GPU is the most powerful device tested while the MPSoC performance is two order of magnitude lower. That being a somewhat expected result, the key question to be investigated in the next sections is whether or not such gap is compensated by the lower energy consumption of MPSoC.

7 Power Consumption measurements

In this section we discuss our work to estimate and compare the instantaneous power, the total energy consumption, the execution time and the energetic cost for a simulation using the Hy-Nbody code for the various devices listed in Table 3.

We also estimate the energy impact of our code in terms of Energy Delay Product (EDP). The EDP proposed by Cameron [6], is a "fused" metric to evaluate trade-off between time-to-solution and energy-to-solution. The EDP is defined as:

$$EDP = E \times T^w \tag{1}$$

where E is the total energy consumed during the run, T is the time-to-solution and w is a parameter to weight performance versus power. Common value of this parameter are $w = 1, 2, 3$ ($w = 3$ was suggested by Cameron), the larger is w the greater the value of performance.

As discussed in Section 3.2, the *Evaluation* kernel is the most computational demanding part of our code, it is strongly compute-bound on every architecture so it is excellent to make energy tests. Relying on these profiling results, we measure the energy consumption during the execution of the *Evaluation* kernel, in an infinite while loop.

In order to minimize the inaccuracies in estimating the current consumed by the CPU and the GPU while running the kernel, we apply two different methodologies, one for the Firefly KR-3399 and one for the Intel desktop with Nvidia GPU.

The Firefly RK-3399 board has been powered by a DC power supply (a Keysight E3634A) to avoid the power draw by the AC-to-DC transformer, which makes the readings more noisy and spread out. 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.

$I^{device}_{impl,baseline}$ is the current consumed by the system running a given code implementation using a particular device (CPU or GPU).

I^{device}_{impl} is the current that we are interested in:

$$I^{device}_{impl} = I^{device}_{impl,baseline} - I_{baseline} \tag{2}$$

$I^{device}_{impl,baseline}$ and $I_{baseline}$ are the mean values over a range of three minutes.

The energy consumed by a given implementation of the kernel (energy-to-solution) is

$$E^{device}_{impl} = V \times I^{device}_{impl} \times T^{device}_{impl} \tag{3}$$

where V and T^{device}_{impl} are the voltage and the kernel running time (time-to-solution averaged over ten runs), respectively (voltage is constant, namely $V =$ 12 Volt).

On the Intel desktop we set the frequency governor to performance level and the electric power draw is measured by means of a power meter (Yokogawa WT310E).

After booting up the platform, we measure the watts hours consumed in idle during a period of three minutes, giving us the $W_{baseline}$ of the system. $W^{device}_{impl,baseline}$ 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 i30ss).

The watts hours (energy-to-solution) that we are interesting in are:

$$W^{device}_{impl} = (W^{device}_{impl,baseline} - W_{baseline}) \times T^{device}_{impl} / \Delta T_3 \tag{4}$$

where T^{device}_{impl} is the kernel running time (time-to-solution averaged over ten runs).

7.1 Experimental setup

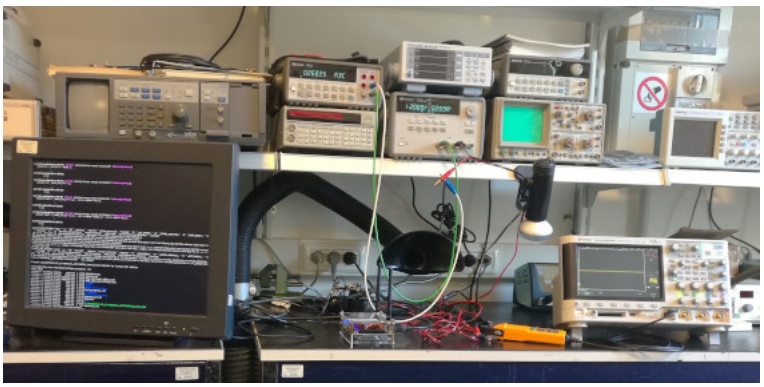

Fig. 5: Experimental setup at the Astronomical Observatory of Trieste - electronic laboratory.

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;
  • 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 450 measurements taken at each run with a sample rate of 2,5 Hz (1 sample every 400 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.

– 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.

  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. Also in this case, the oscilloscope and the current clamp were used to monitor the dynamic behaviour of the current consumption, but this was possible only with a limited subset of the tests, since only the auxiliary power supply input of the GPU could be intercepted. The discrete GPU is also supplied by the PCIe connector and thus the measurements taken with the current clamp are not reliable.

Table 3: The main characteristics of the configuration used in the test including the arithmetic's capacity of the accelerators. The last two columns are the device energy measured in [Watt/h]. The $\mathbf{E_{baseline}}$ is the idle energy on 3 minutes of evaluation and the $\mathbf{E_{impl}^{device}}$ is the energy for 3 minutes of Hy-Nbody kernel continuous execution.

Device	**Core Workgroup**	**Arithmetic**	**DP/FP**	**EX/FP**	$\mathbf{E_{baseline}}$	$\mathbf{E_{impl}^{device}}$
					[Watt/h]	
`i7-3770`	4	DP	1		3.95	6.92
`Arm A53`	4	DP	1		0.15	0.23
`Arm A72`	2	DP	1		0.15	0.34
`Arm Mali-T864`	64	EX	1/10	1/5[4]	0.15	0.29
`GeForce-GTX-1080`	64	EX	1/32	5/8	3.95	8.40

[4] Arm Optimized

8 Energy results

Energy-to-solutions and time-to-solutions are obtained running the *Evaluation* kernel using both DP and EX arithmetic using 65536 particles. That number of particles has been chosen in order to keep busy the device for a reasonable amount of time and therefore to make robust measurements.

For each device we estimate both the $\mathbf{E_{baseline}}$ and the $\mathbf{E^{device}_{impl}}$, i.e. the energy for 3 minutes of Hy-Nbody kernel continuous execution. Our energy consumption measurements are reported in Table 3. Not surprisingly, the most

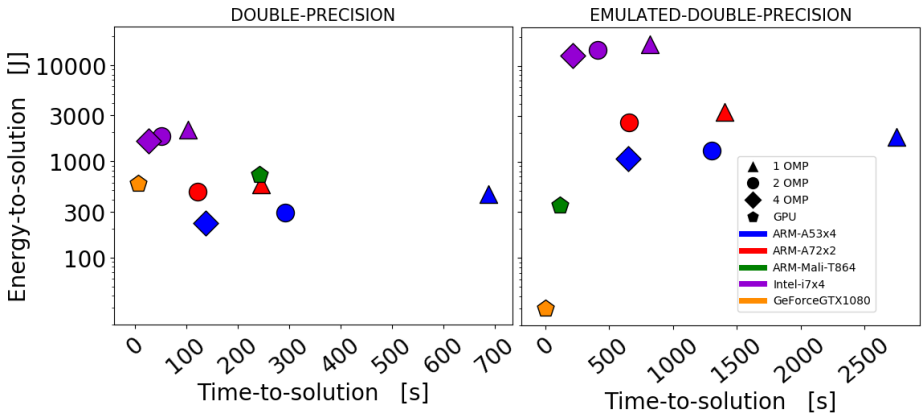

Fig. 6: Energy-to-solution (in Joule) as a function of time-to-solution (in second) for DP (left panel) and EX (right panel) arithmetic. Blue symbols for Arm-A53x4 CPU, red symbols for Arm-A72x2 CPU, green symbol for Arm Mali-T864, violet symbols for Intel-i7x4 CPU and orange symbol for Nvidia GeForce-GTX-1080. Triangle up for 1 OMP thread (serial calculation), circle for 2 OMP threads, diamond for 4 OMP threads, pentagon for GPUs with work-group size of 64.

energy consuming devices are the i7 and the Nvidia GPU that absorb more than twenty times the Firefly-RK3399.

Time-to-solution and energy-to-solution results are plotted in Figure 6 for both DP and EX arithmetic.

The most effective device, both in terms of time-to-solution and energy-to-solution, is the dedicated 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 4 times more power-efficient than the single-core Intel-i7, running at 3.40 GHz. Moreover, CPUs do not benefit at all of the EX arithmetic. Indeed, their performances degrade when using EX (note that

right and left panels in Figure 6 have different x-scales): for this reason on Figure 7 we plot the "best results" in terms of computational performances, namely DP arithmetic for CPUs and EX arithmetic for GPUs.

When using EX arithmetic, the performances are improving dramatically for the GPUs.

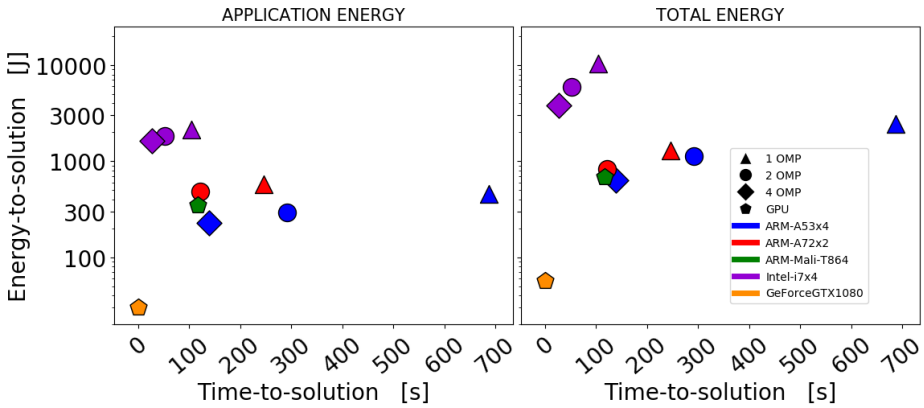

Fig. 7: Energy-to-solution (in Joule) as a function of time-to-solution (in second) for best configuration i.e. DP for CPUs and EX for GPUs. On the left panel we plot the energy-to-solution for the running kernel excluding the energy baseline, on the right panel we plot the total electric power drawn by the system running the kernel. Symbols and lines are the same as described in Figure 6.

As we are also interested in the total energy impact (including boards base line), this is the real energy consumption of the platforms when making computations. In Figure 7 we present the best cases plot for the total energy (right panel). The effect of the higher baseline is affecting mainly the computations on i7 and GPGPU, while the CPUs and GPUs on the Firefly have roughly the same power consumption and energy behaviour.

Finally we compare our devices in terms of EDP. In Figure 8 we plot the EDP for 3 values of the w parameter comparing the different devices. For the CPUs we vary the number of cores involved in the calculation, while for GPUs we fixed the of work-group size (no significant energy difference has been noticed modifying the work-group size).

As expected, for CPUs the EDP factor is a function of the cores used, the most is the occupancy of the CPU the best is the energy impact. When performances are highly valued, the Intel processors and the GTX-1080 GPU are the devices with best trade-off between time-to-solution and energy-to-solution (in particular the GTX-1080 GPU), even if their instantaneous power is higher than the Firefly-RK3399 board.

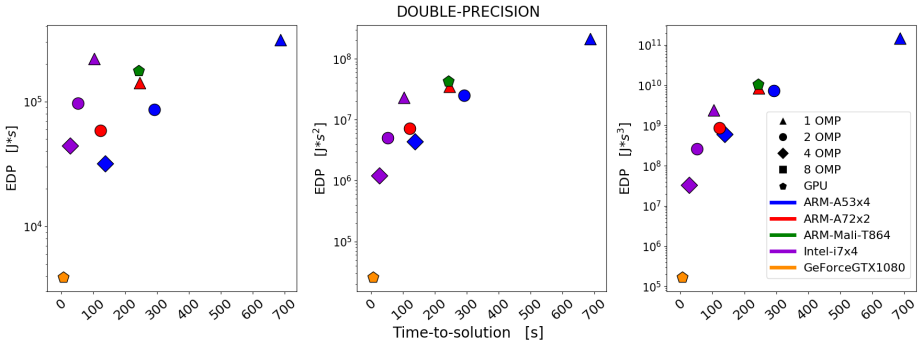

Fig. 8: EDP as a function of time-to-solution (in second) for double-precision arithmetic varying the weight of the execution time ($w = 1, 2, 3$). Symbols and lines are the same as described in Figure 6.

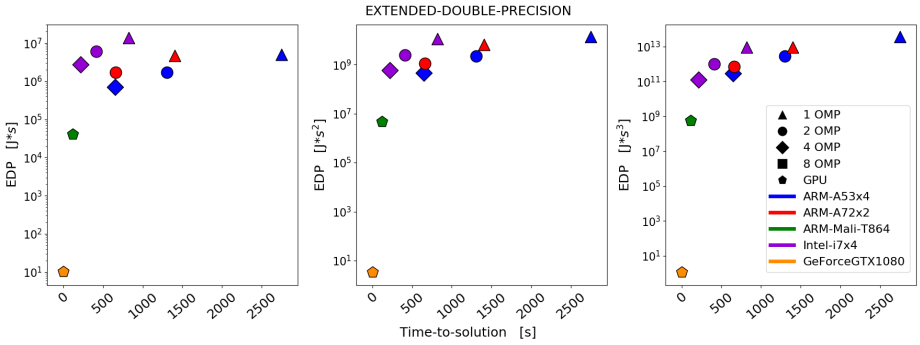

Fig. 9: EDP as a function of time-to-solution (in second) for emulated double-precision arithmetic varying the weight of the execution time ($w = 1, 2, 3$). Bottom-left is better. Symbols and lines are the same as described in Figure 6.

On Figure 9 we measure the EDP when using the EX arithmetic. In this case both the Mali GPU and the GTX-1080 are favoured in terms of energy and computing time as also shown in Figure 7.

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.

9 Conclusion 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 the energy-to-performance ratio. In this work, we begin to explore the impact of the software design of a scientific application on its energy-to-solution and time-

to-solution footprints exploiting low-cost SoC-based platforms. We compare the relative power efficiency between the CPU (single-core, dual-core, multi-core) and the GPU on SoC using a real scientific application.

The code we used is conceived to minimize the MPI communications, so that we could evaluate the impact of the computation, on both GPUs and CPUs, on the power consumption. Given the negligible role of networking and the absence of I/O, our results are sufficiently robust for the discussion and the conclusions that we present. Hence, they can be considered representative of arithmetic-intensive scientific codes with a small need of MPI communication.

Furthermore, the effectiveness of EX arithmetic to exploit commercial gaming-class GPUs in scientific calculations is also a significant outcome of this work that may have a general interest.

Therefore, we identify an application able to exploit both the CPUs and GPUs, and then we study the electric power consumption at specific instant during a run.

We use the Firefly-RK3399 Arm MPSoC based board that meets the requirements identified to successfully run a scientific HPC code in particular: 4GB of RAM and excellent expansion capacity. The results obtained with this board are compared with the ones from a general purpose x86 desktop and PCIe GPGPUs, the Nvidia GeForce-GTX-1080. We deliberately use a consumer grade desktop and a commodity GPGPU to compare "similar" platforms (single boards computers are not designed to compare with high-end HPC servers).

We analyze how to optimize our code to benefit of heterogeneous platforms with a big.LITTLE architecture and integrated GPU. This is complicated mainly by two problems: the efficient scheduling of the algorithms on big.LITTLE architecture and the use of full double precision arithmetic in order to provide real scientific results. The first affects the way a developer can run a HPC code on the MPSoC, the latter involves the efficient use of consumer grade GPUs.

Our results show that even if the energy consumption of the single board computers is orders of magnitude lower that the one form a desktop (and then also lower than a server), when evaluating the energy-to-solution of an HPC designed core, the best device seems to be the Nvidia GeForce-GTX-1080 at least for this specific code. However, also the low power Mali embedded GPU performs much better even of the i7, as also demonstrated by the EDP analysis.

Even if not comparable with the GPGPU, the overall single board computer (CPUs and GPUs) is extremely promising both in terms of performances and energy consumption, in particular if applications will be able to use at the same time the two CPU clusters and the associated GPU. This requires to re-engineer the applications completely.

Furthermore, a crucial finding of this work is the effect of the emulated double precision on the GPU performances. Our tests using DP arithmetic have demonstrated the poor results of those devices. This is not an unexpected behaviour, since consumer grade GPUs (and also MPSoC Mali GPUs) are not built for high performance DP. This is because they are targeted towards games and game developers. So vendors commonly do not cram DP compute cores in their

GPUs. However the introduction of EX arithmetic highly improves the performance filling the gap with the SP capability and opening the path for a success full and cheap use of those devices also on HPC area.

In conclusion, we have shown that SoC technology is emerging as a promising alternative to "traditional" technologies for HPC, which are more focused on peak-performance than on power-efficiency. That being especially true when a significant effort was spent in re-engineering the computaitonal parts and the communication schemes, since it is expected that to achieve small enough times-to-solution a large number of low-power-consumption devices will be needed.

How a more complex communication pattern may affect the time-to-solution and energy-to-solution is not a focus of this preliminary work, although that is obviously a major point in HPC. We leave this to a forthcoming work in which we use different, more complex codes that implements a large variety of different algorithms. For instance, a tree-code - suited for a different class of problems than a direct N-Body - would challenge quite differently both the cache hierarchy and the interconnect. Also, on one hand, the implementation of tree-related routines for a GPU are far more efficient than a simple vector-based series of instructions, and on the other hand the execution pattern of more complex codes that include more physical modules, will change heavily the role balance between CPU and GPU.

This analysis has been done using an Astrophysical code, however our results are valid also for any computing intensive code that requires double precision arithmetic: in practice the majority of codes in science. Furthermore, the use of EX arithmetic to exploit commercial GPUs is an important result that opens to the possibility to use gaming GPUs also for HPC.

Our future plan is to assess the energy footprint of other aspects of this application, such as network and I/O and compare clusters of MPSoCs with HPC resources, where multi-node MPI communication becomes an important aspect of a simulation.

10 Acknowledgments

This work was carried out within the ExaNeSt (FET-HPC) project (grant no. 671553), the ASTERICS project (grant no. 653477) and EuroExa (FET-HPC) project (grant no. 754337) funded by the European Unions Horizon 2020 research and innovation programme.

References

1. Ammendola R., Biagioni A. , Cretaro P., Frezza O., Cicero FL et al.: The Next Generation of Exascale-Class Systems: The ExaNeSt Project. In Euromicro Conference on Digital System Design (DSD), Vienna, pp. 510-515 (2017) http://dx.doi.org/10.1109/DSD.2017.20
2. Arm Mali GPU OpenCL Developer Guide, Version 3 (2016) http://infocenter.arm.com/help/topic/com.arm.doc.100614_0300_00_en/arm_mali_gpu_opencl_developer_guide_100614_0300_00_en.pdf

3. Gaster B., Howes L.W., Kaeli D.R., Mistry P., and Schaa D.: Heteroge- neous Computing with OpenCL - Revised OpenCL 1.2 Edition. Morgan Kaufmann, (2013)
4. Berczik P., Nitadori K., Zhong S., Spurzem R., Hamada T., Wang X., Berentzen I., Veles A., Ge W.: High performance massively parallel direct N-body simulations on large GPU clusters. In: International conference on High Performance Computing, Kyiv, Ukraine, October 8-10, p. 8-18, pp 818 (2011)
5. Bonomi F. , Milito R., Zhu J., Addepalli S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing - MCC 12, ACM Press, New York, New York, USA, p. 13 (2012) http://dx.doi.org/10.1145/2342509.2342513
6. Cameron K.W., Ge R., Feng X., Varner D., Jones C.: High-performance, power-aware distributed computing framework. In Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC), ACM/IEEE, (2004)
7. Capuzzo-Dolcetta R., Spera M.: A performance comparison of different graphics processing units running direct N-body simulations. Computer Physics Communications 184:25282539 (2013)
8. Doucet K., Zhang J.: Learning cluster computing by creating a Raspberry Pi cluster. In: Proceedings of the SouthEast Conference, in: ACM SE 17, pp. 191194. (2017). http://dx.doi.org/10.1145/3077286.3077324
9. Durand Y., Carpenter P. M., Adami S., Bilas A., Dutoit D., et al.: EUROSERVER: Energy Efficient Node for European Micro-Servers. In 17th Euromicro Conference on Digital System Design, Verona, pp. 206-213 (2014) doi:10.1109/DSD.2014.15
10. Farber R.: Parallel Programming with OpenACC (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2016)
11. Goz D., Tornatore L., Bertocco S., Taffoni G.: Direct N-body code designed for heterogeneous platforms. In: INAF-OATs technical report, 223, July (2018) doi:10.20371/INAF/PUB/2018_00002.
12. Harfst S., Gualandris A., Merritt D., Spurzem R., Portegies Zwart S., Berczik P.: Performance analysis of direct N-body algorithms on special-purpose supercomputers. New Astronomy 12:357377 (2007)
13. Katevenis M., Chrysos N., Marazakis M., Mavroidis I., Chaix F., Kallimanis N., et al.: The ExaNeSt Project: Interconnects, Storage, and Packaging for Exascale Systems, 2016 Euromicro Conference on Digital System Design (DSD), Limassol, pp. 60-67 (2016)
14. Katevenis M.,Ammendola R., Biagioni A., Cretaro P., Frezza O., Lo Cicero F., et al.: Next generation of Exascale-class systems: ExaNeSt project and the status of its interconnect and storage development, Microprocessors and Microsystems, Volume 61, Pages 58-71 (2018)
15. Keller M., Beutel J., Thiele L.: Demo Abstract: MountainviewPrecision Image Sensing on High-Alpine Locations. In: D. Pesch, S. Das (Eds.), Adjunct Proceedings of the 6th European Workshop on Sensor Networks, EWSN, Cork, pp. 1516 (2009)
16. H. Kobayashi:Feasibility Study of a Future HPC System for Memory-Intensive Applications: Final Report. In: Resch M., Bez W., Focht E., Kobayashi H., Patel N. (eds) Sustained Simulation Performance 2014. Springer, Cham (2014)
17. Kogge P., Bergman K., Borkar S., Campbell D., Carson W., Dally W., Denneau M., Franzon P., Harrod W., Hill K., et al.: Exascale computing study: technology challenges in achieving exascale systems. Tech. rep., University of NotreDame, CSE Dept. (2008)
18. Konstantinidis S., Kokkotas K.: MYRIAD: a new N-body code for simulations of star clusters. Astronomy and Astrophysics 522:A70 (2010)

19. Mantovani F., Calore E.: Performance and Power Analysis of HPC Workloads on Heterogeneous Multi-Node Clusters. In: Journal of Low Power Electronics and Applications - Volume 8, n.2 (2018) http://www.mdpi.com/2079-9268/8/2/13
20. Martinez K.,Basford P.J., DeJager D., Hart J.K.: Using a heterogeneous sensor network to monitor glacial movement. In: 10th European Conference on Wireless Sensor Networks, Ghent, Belgium, (2013).
21. Nitadori K., Aarseth S.J.: Accelerating NBODY6 with graphics processing units. MNRAS 424:545552 (2012)
22. Nitadori K., Makino J.: Sixth- and eighth-order Hermite integrator for N-body simulations. New Astronomy 13:498507, (2008)
23. Nickolls J., Buck I., Garland M., Skadron K.: Scalable Parallel Programming with CUDA. Queue 6, 2 40-53, (2008), https://doi.org/10.1145/1365490.1365500
24. Ou, Z., Pang, B., Deng, Y., Nurminen, J., Yla-Jaaski, A., and Hui, P.:Energy- and cost-efficiency analysis of ARM-based clusters. In: 12th IEEE/ACM InternationalSymposium on Cluster, Cloud and Grid Computing, CCGrid 2012, pp. 115–123 (2012)
25. Rajovic N.,Rico A.,Puzovic N.,Adeniyi-Jones C.,Ramirez A.: Tibidabo:Making the case for an ARM-based HPC system. Future Gener. Comput. Syst. 36 322334 (2014) http://dx.doi.org/10.1016/J.FUTURE.2013.07.013
26. Spera M.: Using Graphics Processing Units to solve the classical N-body problem in physics and astrophysics. ArXiv e-prints 1411.5234 (2014)
27. Spera, M., & Capuzzo-Dolcetta, R.: Rapid mass segregation in small stellar clusters. In Astrophysics and Space Science, Volume 362, Issue 12, article id.233, 12 (2017)
28. Terpstra D., Jagode H., You H., Dongarra J.: Collecting performance datawith papi-c. In: M. S. Muller, M. M. Resch, A. Schulz, W. E. Nagel (Eds.),Tools for High Performance Computing 2009, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 157173 (2009)
29. Thall A.: Extended-precision floating-point numbers for gpu computation. p 52, (2006) DOI 10.1145/1179622.1179682
30. Turton P., Turton T.F.: Pibraina cost-effective supercomputer for educational use. In: 5th Brunei International Conference on Engineering and Technology, BICET 2014, pp. 14, (2014)
31. Upton E.,Halfacree G.: RaspberryPi UserGuide, fourthed. Wiley (2016)
32. Yoneki E.: Demo:RasPiNET:decentralised communication and sensing platform with satellite connectivity. In: Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks - CHANTS 14, ACM Press, New York, New York, USA, pp. 8184 (2014) http://dx.doi.org/10.1145/2645672.2645691