Hindawi
Scientific Programming
Volume 2017, Article ID 7206595, 14 pages
https://doi.org/10.1155/2017/7206595

Research Article

Power-Efficient Computing: Experiences from the COSA Project

Daniele Cesini,[1] Elena Corni,[1] Antonio Falabella,[1] Andrea Ferraro,[1] Lucia Morganti,[1] Enrico Calore,[2] Sebastiano Fabio Schifano,[2] Michele Michelotto,[3] Roberto Alfieri,[4] Roberto De Pietri,[4] Tommaso Boccali,[5] Andrea Biagioni,[6] Francesca Lo Cicero,[6] Alessandro Lonardo,[6] Michele Martinelli,[6] Pier Stanislao Paolucci,[6] Elena Pastorelli,[6] and Piero Vicini[6]

[1]*CNAF-Italian Institute for Nuclear Physics (INFN), Bologna, Italy*
[2]*University of Ferrara and Italian Institute for Nuclear Physics (INFN), Ferrara, Italy*
[3]*Italian Institute for Nuclear Physics (INFN), Padova, Italy*
[4]*University of Parma and Italian Institute for Nuclear Physics (INFN), Parma, Italy*
[5]*Italian Institute for Nuclear Physics (INFN), Pisa, Italy*
[6]*Italian Institute for Nuclear Physics (INFN), Roma, Italy*

Correspondence should be addressed to Enrico Calore; enrico.calore@fe.infn.it

Received 5 February 2017; Revised 10 July 2017; Accepted 27 July 2017; Published 25 September 2017

Academic Editor: Basilio B. Fraguela

Energy consumption is today one of the most relevant issues in operating HPC systems for scientific applications. The use of unconventional computing systems is therefore of great interest for several scientific communities looking for a better tradeoff between time-to-solution and energy-to-solution. In this context, the performance assessment of processors with a high ratio of performance per watt is necessary to understand how to realize energy-efficient computing systems for scientific applications, using this class of processors. Computing On SOC Architecture (COSA) is a three-year project (2015–2017) funded by the Scientific Commission V of the Italian Institute for Nuclear Physics (INFN), which aims to investigate the performance and the total cost of ownership offered by computing systems based on commodity low-power Systems on Chip (SoCs) and high energy-efficient systems based on GP-GPUs. In this work, we present the results of the project analyzing the performance of several scientific applications on several GPU- and SoC-based systems. We also describe the methodology we have used to measure energy performance and the tools we have implemented to monitor the power drained by applications while running.

1. Introduction and Related Works

Energy consumption has increasingly become one of the most relevant issues for scaling up the performances of modern HPC systems and applications, a trend which is expected to continue in the foreseeable future. This implies that costs related to the running of applications are more and more dominated by the electricity bill, and for this reason the adoption of energy-efficient processors is necessary, ranging from many-core processors architectures like *Graphics Processing Unit* (GPU) to *System on Chip* (SoC) designed to meet the demands of the mobile and embedded market. SoC hardware platforms typically embed in the same die low-power multicore processors combined with a GPU and all the circuitry needed for several I/O devices. These processors feature high performance-per-watt ratio aiming at high energy-efficiency but at the same time require careful programming and optimization to be also compute-efficient. Moreover, for the case of off-the-shelf SoCs, various limitations may arise: 32-bit architectures, small CPU caches, small RAM sizes, high latency interconnections, unavailability of ECC memory, and so forth.

Investigating and assessing the performance of these systems for scientific workloads are the aim of the *Computing On SOC Architecture* (COSA) project [1], a 3-year initiative that is funded by the *Italian Institute for Nuclear Physics* (INFN) and started in January 2015. Seven INFN departments have been involved in the COSA project, namely, CNAF (Bologna),

Pisa, Padova (Padua), ROMA1 (Rome), Ferrara, Parma, and Legnaro National Laboratories (LNL).

Processors based on ARM architecture have recently attracted a strong interest from several research communities as energy-efficient building blocks for HPC clusters [2], microservers, and other computing systems. They are widely adopted in several commercial low-power and battery-powered devices, such as tablets and smartphones, and many SoCs embedding ARM cores are designed to have, as main strength points, low power consumption and high energy-efficiency. Several research projects have then investigated towards this direction. Among them, the Mont-Blanc project [3, 4], coordinated by the Barcelona Supercomputing Center [5], has deployed several generations of HPC clusters based on ARM processors, developing also the corresponding ecosystem of HPC tools targeted to this architecture. Another project along this direction is the EU-FP7 EUROSERVER [6], coordinated by CEA [7], which aims to design and prototype technology, architecture, and systems software for the next generation of datacenter "microservers," exploiting 64-bit ARM cores.

Many SoCs based on ARM cores embed GPUs in the same die, and for this reason several projects aim also to exploit these GPUs' computing power [8, 9]. One of such projects, particularly interesting for adopting off-the-shelf boards, is the ICARUS project [10], aiming to build and study the performance of a mobile solar-powered cluster of NVIDIA Jetson boards.

Other research groups are exploring Dynamic Voltage and Frequency Scaling (DVFS) techniques as a way to modulate power consumption of processor and memory, scaling the clock frequency of one or both subsystems according to the execution of memory- or compute-bound application kernels [11]. More recently, other projects are focusing also on Near Threshold Voltage (NTV) computing [12], making the processor work at even lower voltages; since this may lead to computation errors, appropriate checks and recomputations have to be added to algorithms in this case.

The COSA project is on the same research line of the above initiatives, aiming to explore the energy-efficiency of several systems, including ARM-based architecture, low-power SoCs, and also multicore and many-core based processors like Intel Xeon CPUs and NVIDIA Tesla GPUs. In addition to strong hardware agnosticism, the COSA project is characterized by being strongly application-driven, aiming to study the energy consumption behavior of a wide set of benchmarks and software, widely used within the INFN.

For this reason, the COSA project shows affinities also with other initiatives, such as the SpiNNaker (Spiking Neural Network Architecture) project [13] proposed by the Advanced Processor Technologies Research group at the University of Manchester [14], the EU FET-HPC project ExaNeSt [15], the INFN APEnet+ project [16], and the INFN-COKA (Computing on Knights Architecture, [17]) project, of which COSA is a natural extension.

In this work, we explore the performance of energy-efficient systems based on multicore GPUs, low-power CPUs, and SoCs. We have ported several scientific workloads and we have investigated the computing and energy performance comparing them with traditional systems mainly based on x86 multicores. We have also evaluated the benefits of manual clock frequency tuning exploiting DVFS (Dynamic Voltage and Frequency Scaling) with respect to the default frequency governors, looking for an optimal tradeoff between energy-to-solution and time-to-solution.

This work is organized as follows. In Sections 2 and 3, we describe the clusters built and maintained by the COSA collaboration and the power measurement tools. In Sections 4 and 5, we report on the benchmarking activities and on the real-life applications ported to SoCs and many-core systems. The paper concludes with concluding remarks and a prospect on future work in Section 6.

2. The COSA Clusters

The COSA project built and currently maintains three computing clusters, located at the departments of CNAF (Bologna), ROMA1 (Rome), and Padova (Padua). A fourth cluster has been installed in Ferrara with the main contribution of the University of Ferrara. The cluster hosted at CNAF is composed of development boards powered by state-of-the-art low-power SoCs with both ARM and Intel architectures. The cluster located in Rome exploits the last-generation FPGAs to prototype low-latency network connections between low-power CPUs. Finally, the third and fourth clusters, located in the Padova and Ferrara departments, are based on traditional, high-end CPUs, accelerators, and network connections. These latter clusters are used as a reference for the performance of the scientific applications run on the other machines. Clearly, comparing high-end servers (equipped with multiple sockets, redundant power supplies, fans, disks, and huge RAM amounts) with stand-alone boards powered by a single SoC is somehow "unfair," but, nevertheless, indications about the limitations and capabilities in terms of energy-efficiency of these low-power systems can be drawn. In the following subsections, we describe these clusters in more detail.

2.1. The Low-Power Cluster Based on SoCs. The CNAF department hosts an unconventional cluster of ARMv7, ARMv8, and x86 low-power SoCs nodes, interconnected through 1 Gbit/s and 10 Gbit/s Ethernet switches. These platforms are used as a testbed for synthetic benchmarks and real-life scientific applications in both single-node fashion and multinode fashion (see Sections 4 and 5).

Ubuntu is installed on all the platforms. A master server is used as a monitoring station and an external network file system hosting all software and datasets is mounted on every cluster node. CPU frequency scaling is used by the Linux operating system to change the CPU frequency for saving power depending on the system load, and the recommended "ondemand" governor is enabled by default. We set governor to "performance" so to avoid dynamic CPU frequency scaling and maximize CPU performance. The GPU frequency has been set to its maximum value (see Table 1).

The ARM cluster is composed of eight NVIDIA Jetson TK1 boards, four NVIDIA Jetson TX1 (64-bit) boards, two ODROID-XU3 boards, a CubieBoard, a SABRE board, and

Table 1: Hardware specifications of the boards in the COSA cluster at INFN-CNAF. Top: ARM; bottom: Intel.

Platform	CPU	GPU	RAM
Freescale SABRE board (i.MX6Q SoC)	4xA9 1.2 GHz	Vivante GC2100 600 MHz	2 GB
Hardkernel ODROID-XU-E (Exynos5 5410)	4xA15 + 4xA7 1.6 GHz	PowerVR SGX544 384 MHz	2 GB
Hardkernel ODROID-XU3 (Exynos5 5422)	4xA15 + 4xA7 2 GHz	ARM Mali-T628 533/695 MHz	2 GB
HiSilicon Kirin 6220	8xA53 1.2 GHz	ARM Mali-450 MP4 700 MHz	1 GB
Hardkernel ODROID-XU3 (Exynos5 5422)	4xA15 + 4xA7 2 GHz	ARM Mali-T628 533/695 MHz	2 GB
Arndale Octa (Exynos 5420)	4xA15 + 4xA7 1.7 GHz	ARM Mali-T628 533/695 MHz	1 GB
NVIDIA Jetson TK1	4 cores +1 2.3 GHz TDP 10 W	NVIDIA Jetson TK1 192 Kepler cores 852 MHz	2 GB
NVIDIA Jetson TX1	4 cores 1.73 GHz TDP 10 W	NVIDIA Jetson TX1 256 Maxwell cores 998 MHz	4 GB
Intel Avoton C2750	8 cores 2.4 GHz TDP 20 W	—	16 GB
Intel Xeon D-1540	8 cores 2.6 GHz TDP 45 W	—	16 GB
Intel Pentium N3700	4 cores 2.4 GHz TDP 6 W	HD graphics	16 GB

an Arndale Octa board, all interconnected with standard 1 Gbit/s Ethernet. We notice that the Jetson TX1 cluster is connected with 1 Gbit/s Ethernet even though a USB-Ethernet bridge provides the physical connection with the SoC, increasing the latency with respect to standard 1 Gbit/s Ethernet connections.

The 64-bit x86 cluster is composed of four mini-ITX boards powered by the Intel Avoton C2750, four mini-ITX motherboards based on the Intel Xeon D-1540 CPU, and four mini-ITX boards based on the Intel Pentium N3700 processor. The "Avoton," the "Xeon D," and the "TX1" clusters are connected with both 1 Gbit/s and 10 Gbit/s Ethernet connections, and the "N3700" processor only with 1 Gbit/s Ethernet network. The 1 Gbit/s connections are provided by on-board connectors, while the 10 Gbit/s links are obtained with a PCI Host Bus Adapter (HBA).

Table 1 summarizes and compares the relevant features of systems available at CNAF. The Thermal Design Power (TDP) of the SoCs in this cluster, when declared, ranges from 5 W of Intel Pentium N3700 to 45 W of the 8-core Intel Xeon D-1540 Processor.

2.2. The High-End Hardware Clusters. At CNAF department, the traditional reference architecture is a x86 node from an HPC cluster, equipped with two Intel Xeon E5-2620 v2 CPUs, 6 physical cores each, HyperThread enabled (i.e., 24HT cores in the single node), and with a NVIDIA K20 GPU accelerator card with 2880 CUDA cores. The HPC server is rated with a TDP of 160 W for the two CPUs (about 80 W each) and 235 W for the GPU.

In Ferrara, another commodity HPC reference cluster named COKA is available (Computing On Kepler Architecture) [18]. The cluster is made of 5 computing nodes, hosting each 2x Intel Xeon E5-2630 v3 CPUs and 8x NVIDA K80 dual-GPU boards, interconnected with Mellanox MT27500 Family [ConnectX-3] InfiniBand HCA (two per node). The TDP of each CPU is 85 W, while for an NVIDIA K80 board it is 300 W, amounting for a total maximum power of 3.2 kW for each computing node or 16 kW for the whole cluster. On this cluster, the power drain can be measured from node's power supplies and read out at ≈1 s granularity thanks to the IPMI (Intelligent Platform Management Interface) protocol. Otherwise, power drain could be monitored also via processor hardware registers, thanks to the PAPI Library [19], for both the Intel CPUs and NVIDIA GPUs, as detailed in Section 3.

2.3. The FPGA-Based Cluster. In the last few years, reconfigurable devices characterized by complex architecture emerged as an effective and powerful alternative to low-power SoC. Last-generation high-end FPGAs, in addition to huge amount of user-programmable logic, include multiple

embedded ARM 64-bit cores (ARM Cortex-A57) running at 1.5 GHz with high-speed interfaces to storage (USB, SATA) and network (10–40 Gb/s Ethernet) standard protocols and tightly coupled to (up) 1 TFLOPS of configurable DSP blocks. It is well known that, beyond the computing performance of the elementary processor, the main issue limiting the scalability of a massive parallel system is the efficiency of the interconnection architecture.

The department of ROMA1 has launched in the past the APEnet+ [16] project aiming to design a low-latency, high-performance 3D Torus network architecture optimized for scientific computing on GPU-accelerated HPC systems. APEnet+ is embedded in FPGA and it implements powerful host interface (PCIe Gen3 x8), a GPU-dedicated low-latency DMA engine, and a number of custom, high-speed serial channels on Torus side. One of the main limiting factors of network performance in the current implementation of APEnet+ architecture (named V5) is the lacking, in target FPGA device (28 nm process FPGA generation), of a high-performance embedded hard-core processor needed to execute highly demanding network supporting computing tasks (e.g., the Virtual-to-Physical address translation as well as DMAs initialization).

In the framework of COSA, the main goals of the ROMA1 department have been the evaluation of the FPGA embedded ARM cores as (1) atomic computing core for a fine-grained parallel HPC system and (2) powerful microprocessor able to sustain the computing tasks required by network support. In this perspective and in full synergy with the activities of EU FET-HPC project ExaNeSt [15], we procured and assembled a small size (6 computing nodes) FPGA-based computing cluster based on the Xilinx Zynq UltraScale+ [20] development kit produced by Trenz Electronic GmbH, equipped with the XCZU9EG-1FFVC900 that integrates a programmable quad-core ARM Cortex-A53 (64-bit) SoC @1.5 GHz. Targeting this platform, we completed the porting of APEnet+ IP V5 and we have deployed the first running release of such, as an FPGA Zynq-based cluster. This cluster sports a standard 1 Gbit/s Ethernet service network and a preliminary working prototype of a low-latency and high-performances Torus network based on multiple 10 Gbit/s point-to-point links. The current development plans foreseen to design the additional features needed (a) to improve the physical link throughput and the switching performance, (b) to optimize network collective operations, (c) to enhance the network IP resiliency capabilities at extreme-size system scale (exaFLOPS), and (d) to support high-radix network topologies (n-D Torus and Dragonfly). The incremental adoption of the new features and optimizations will allow delivering the final optimized FPGA-based cluster at the end of 2017.

3. Power Monitor Tools

We developed and implemented several fine-grained power monitoring systems that are able to provide power and energy readings for a generic application.

The first of these systems, hereafter *papi*, is a software wrapper [21] that exploits the PAPI Library [19] to read appropriate hardware registers containing power-related information. This wrapper allows applications to directly start and stop measurements using architecture-specific interfaces, such as the *Running Average Power Limit* (RAPL) for Intel CPUs and *NVIDIA Management Library* (NVML) for NVIDIA GPUs. The *papi* system is particularly useful in high-end systems, where processors commonly implement these registers and their usage is well documented.

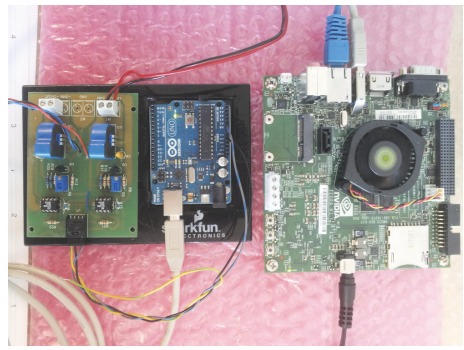

Figure 1: The custom power meter attached to a Jetson TK1.

The second system, hereafter *custom*, requires dedicated hardware [22] and represents a viable solution whenever appropriate hardware registers are unavailable or show difficulties in their readout. For example, this custom power meter can be used to monitor the power drained by SoC development boards, as shown in Figure 1. The *custom* setup uses an analog current-to-voltage converter (LTS 25-NP current transducer) and an Arduino UNO board; the latter uses its embedded 10-bit ADC to digitize current readings and store them in memory. The Arduino UNO board is synchronized with the development board hosting the SoC through a simple serial protocol built over a USB connection. The monitor acquires current samples with 1 ms granularity. This setup is able to correlate current measurements with specific application events with an accuracy of a few milliseconds, minimally disrupting the execution of the application to profile. The application, while running, can start and stop the out-of-band measurement, letting the Arduino UNO board store readings in its memory. When needed (e.g., after the end of a function to energy profile), data can be offloaded from the Arduino UNO board to the application itself.

Another power measurement equipment, hereafter referred to as *multimeter*, consists of a DC power supply, a high-precision Tektronix DMM4050 digital multimeter for DC current measurements connected to National Instruments data logging software, and a high-precision AC power meter. When this monitoring system is used, the AC power of the high-end server node is measured by a Voltech PM300 Power Analyzer upstream of the main server power supply (measuring on the AC cable). Instead, for the SoCs, the DC current was absorbed downstream of the power supply. We believe that such difference does not affect significantly the results, given the close to one cos phi factor of the server power supply.

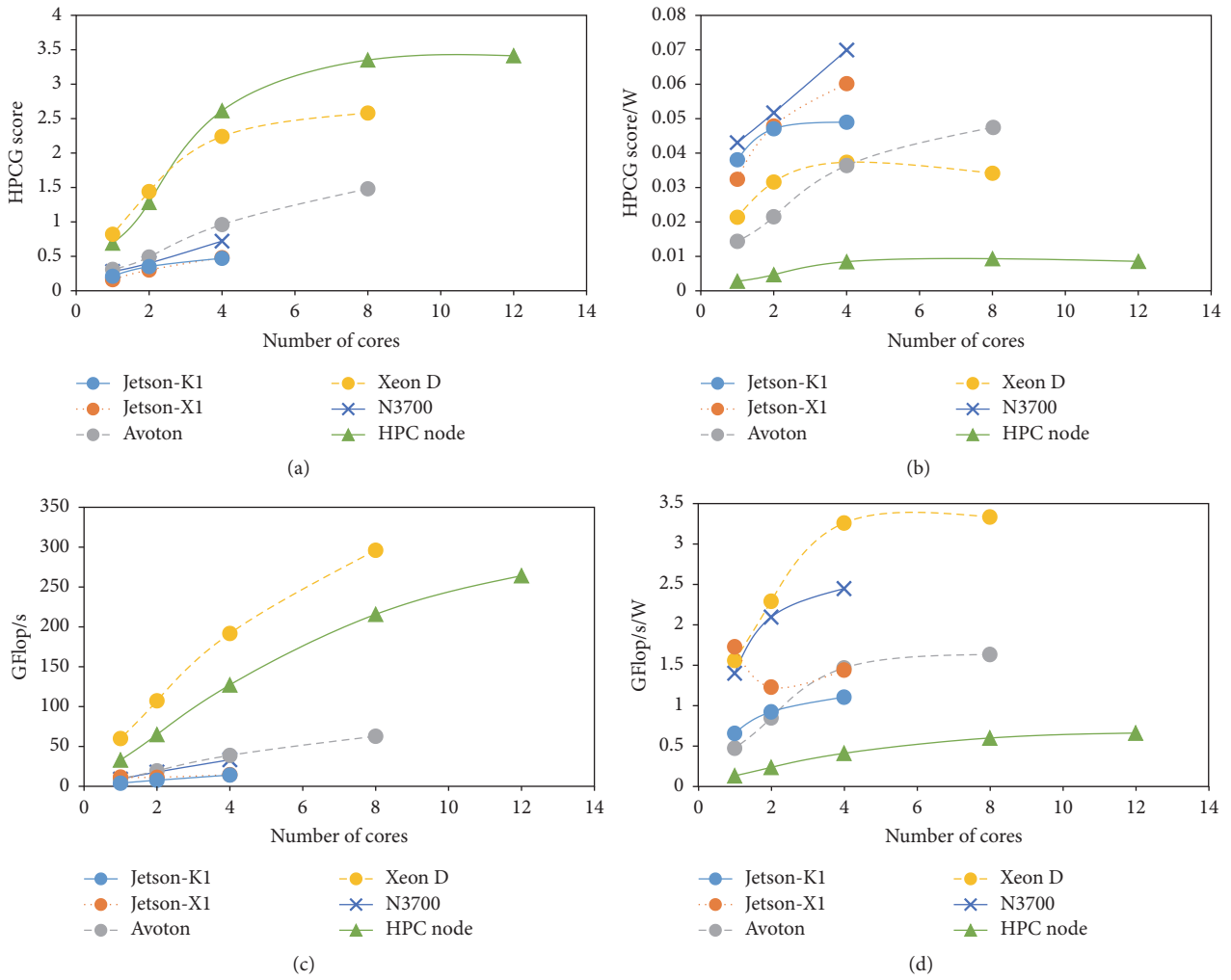

Figure 2: Running two kinds of benchmarks on the various CPUs of the COSA CNAF cluster. (a) and (b) show the results of the HPCG benchmark: HPCG score (weighted GFLOP/s) on (a) and ratio between HPCG score and absorbed power on (b), for increasing number of CPU cores. (c) and (d) show GFLOP/s and power ratio for the home-made MATMUL benchmark.

4. Synthetic Benchmarks

In order to characterize the single-node performance of the various CPUs available in the COSA CNAF clusters, we performed several types of benchmarks: both synthetic tests developed in house and well-documented test suites publicly available in literature.

Here, results are shown only for the most recent and powerful CPUs of the COSA CNAF clusters, and we stress that any measurement of the total power of the cluster is out of the scope of this work. Also, the synthetic benchmarks showed in this section test only CPUs, not GPUs, and we refer the reader to the following section for real-life workloads executed on the GPUs of the COSA cluster.

From the broad spectrum of tests which are handy in the literature, here we show the results obtained for a test based on the High-Performance Conjugate Gradients (HPCG) Benchmark [23], a renowned new metric for ranking HPC systems, designed to exercise computational, communication, and memory access patterns which are frequently observed in real-life applications, for example, with low compute-to-data-access ratios.

In Figures 2(a) and 2(b), results of the HPCG metrics are shown in terms of weighted GFLOP/s ("HPCG score," (a)) and power absorbed during the execution (b), which has been measured using the *multimeter* system described in Section 3.

As is naturally expected, the absolute performances of a high-end server are much higher. However, the situation reverses itself when considering the power ratio.

This holds true also for MATMUL, a home-made test, written in C and parallelized with OpenMP, which performs single-precision matrix multiplication using SGEMM function from the OpenBLAS library [24]. Figures 2(c) and 2(d) show instead the GFLOP/s and absorbed power by MATMUL for a size of the matrices equal to 4096.

We are currently investigating the weird behavior of the Jetson TX1 board when MATMUL is executed in two cores (see Figure 2(d)).

We note that the Jetson TK1 board performs slightly better than the newer Jetson TX1, although 32-bit architecture is a significant limitation and the somewhat higher clock frequency explains its better performances.

In general, Jetson TX1 and N3700 boards, which are already 64-bit platforms, seem to be very promising low-power architectures. Interestingly, we notice that the performances of the Xeon D-1540 platform resemble, and even overtake, in the case of MATMUL, those of a traditional high-end server. The nice performances of the Xeon D-1540 are ascribable to the capacity of the SGEMM function to exploit the FMA instruction, which is missing in the other boards. However, power consumption is higher than the TDP (80 W versus 45 W of TDP), both with and without turbo boost enabled.

We are fully aware that comparing small low-power boards and HPC nodes, usually equipped with high-end GPU boards, several disks, and large RAM, cannot be fully fair; we also know that application spectrum running on HPC nodes is not directly comparable with that running on small development boards. However, we think that this comparison helps to highlight useful information about limits and potentialities of low-power SoCs and processors and gives interesting hints for further investigations in using this class of computing systems for HPC workloads.

The next section illustrates the results obtained with more realistic tests, that is, running on the COSA clusters several applications taken from different realms of science.

5. Applications

5.1. Lattice Boltzmann. Several fields of computational fluid dynamics are increasingly using Lattice Boltzmann Methods (LBM) to study behavior of flows and to solve numerically the equation of motion of flows in two and three dimensions. These methods can be efficiently implemented on computing systems and are also able to handle complex and irregular geometries as well as multiphase flows.

LBM are discrete in position and momentum spaces and are based on the synthetic dynamics of *populations* associated with the edges of a discrete 2D or 3D lattice. At each time step, populations of each site are propagated (i.e., copied from the neighboring sites), and then incoming populations collide among each other, mixing their values (i.e., new populations values are computed from old ones and the neighbors ones). See [25] for a deeper introduction.

LBM are labeled as $DxQy$, where x represents the dimension, 2D or 3D, and y represents the number of populations associated with each lattice site. Here, we consider the state-of-the-art $D2Q37$ model for simulation of two-dimensional fluids with 37 populations per site. This model correctly reproduces the thermohydrodynamic equations of motion of a fluid in two dimensions and also enforces the equation of state for an ideal gas $p = \rho T$ [26, 27].

From a computational point of view, LBM are *easy* to implement and a large degree of parallelism can be exploited, making them suitable to run on state-of-the-art multicore and many-core processors. The most relevant steps performed by a Lattice Boltzmann simulation are the computation of the `propagate` and `collide` functions:

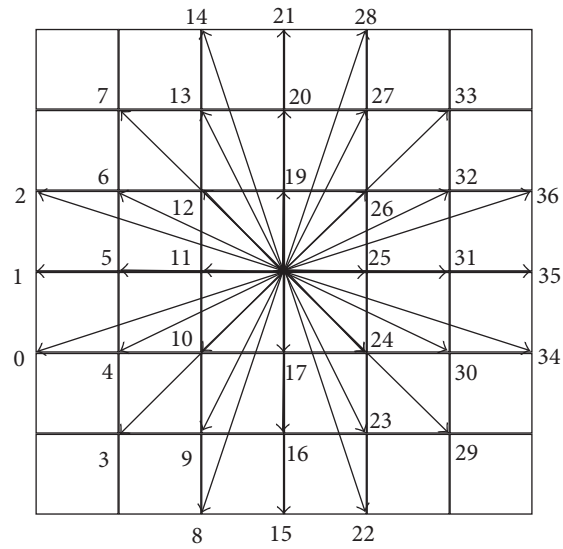

Figure 3: The 37-element stencil used for the computation of `propagate` function in the D2Q37 LB code.

(1) `propagate` moves populations across lattice sites according to the stencil pattern shown in Figure 3; it collects at each site all populations that will interact at the next phase: `collide`. Implementation-wise, `propagate` moves blocks of memory locations allocated at sparse memory addresses, corresponding to populations of neighbor cells.

(2) `collide` performs all the mathematical steps associated with the computation of the collisional operator and computes the population values at each lattice site at the new time step. Input data for this phase are the populations gathered by the previous `propagate` phase.

The D2Q37 LBM is more complex than simpler LB models such as D2Q9 or D2Q17 because the `propagate` function uses a fourth-order scheme of movements, exchanging populations with neighbors up to distance 3 for each site. This translates into severe requirements in terms of memory bandwidth and floating-point throughput. `propagate` implies accessing 37 neighbor cells to gather all populations making this step mainly memory-bound. `collide` requires approximately 7600 double-precision floating point operations per lattice site. `collide` exhibits a significant arithmetic intensity and is the dominating part of the overall computation, taking roughly 90% of the total run-time.

The D2Q37 model has been implemented and extensively optimized on a wide range of parallel machines like BG/Q [28] as well as on a cluster of nodes based on traditional commodity x86 CPUs [29], GPUs [30–32], and Xeon-Phi [33, 34]. It has been extensively used for large-scale production simulations of convective turbulence [35, 36].

Table 2: Best EDP values, with corresponding *energy-to-solution* and *time-to-solution*, running the `collide` kernel in single precision. Lattice: 128 × 1024.

Processor	E_s [J] per iter.	T_s [ms] per iter.	EDP [J s]
GK20A	0.30	42	0.013
Cortex-A15	0.67	58	0.039
Cortex-A53	0.52	77	0.040

Performance and energy requirements of the `propagate` and `collide` functions were estimated on several architectures, measuring, respectively, the time-to-solution (T_s) and the energy-to-solution (E_s).

5.1.1. Low-Power SoCs. Using the *custom* power meter system described in Section 3, T_s and E_s of the `propagate` and `collide` functions have been measured while running on the CPU and on the GPU of a Jetson TK1 [22]. Two different implementations have been used, a plain C one, using ARMv7 *intrinsics* (with ARM Cortex-A15 processor), and a CUDA one (GK20A, the GPU component of the Jetson TK1 SoC), respectively, for the CPU and GPU. Since on this board the clock frequencies of both processors (and memory too) could be changed thanks to DVFS (*Dynamic Voltage and Frequency Scaling*), both T_s and E_s vary according to these frequencies. Several runs have been performed for every possible clock combination showing the frequency optimization space available on this SoC for both processors [22].

The plain C code has been also ported to the ARMv8 architecture and run also on the Cortex-A53 hosted on a HiSilicon Kirin 6220.

Several tradeoff points between E_s and T_s can be identified for each tested processor. Choosing a single metric, for example, the EDP (*Energy Delay Product*), we can select an optimal tradeoff point and perform a comparison according to it, as shown, for example, in Table 2 for the `collide` function.

5.1.2. High-End System. Modern high-end architectures also allow tuning processor clocks through DVFS. We have then changed the clock frequency on computing elements, both CPUs and GPUs, of the COKA cluster and read power values related to processors with *papi* as described in Section 3. We used different code versions, optimized, respectively, for Intel Xeon E5-2630 v3 CPU and for K80 GPUs. For the latter, we ran only on one GPU, out of the two hosted in a NVIDIA Tesla K80 board. In this case, in order to run for a significant amount of time we had to use a larger lattice of 1024 × 8192 points. Selecting best EDP values, we compare the two architectures as shown in Table 3.

We also evaluated the energy-saving potential of DVFS tuning for the full high-end HPC node, running a full Lattice Boltzmann simulation code on 16 GPUs. In this case, we measured the power drain at the wall socket, performing the readout from the power supplies through IPMI (Intelligent Platform Management Interface). Results are reported in Figure 4. As we see, changing the frequency from default to 732 MHz has little to no impact on execution time but allows saving ≈7% of the total energy of the computing node. The reason for this is associated with the computational needs of kernel functions executed by the application [37]. In fact, as a general rule, the throughput of the processor should balance that of the memory, and increasing the processor frequency does not bring any benefit if the application performance is limited by memory accesses. More in general, it is possible to decrease the processor frequency without impacting performances (up to a threshold below of which the application becomes compute-bound) for any code spending significant portions of its execution time in memory-bound operations or waiting for communications and synchronizations.

Table 3: Best EDP values, with corresponding *energy-to-solution* and *time-to-solution*, running the `collide` kernel in double precision. Lattice: 1024 × 8192.

Processor	E_s [J] per iter.	T_s [ms] per iter.	EDP [J s]
Tesla K80	12.4	86.1	1.07
E5-2630 v3	54.6	603.9	32.97

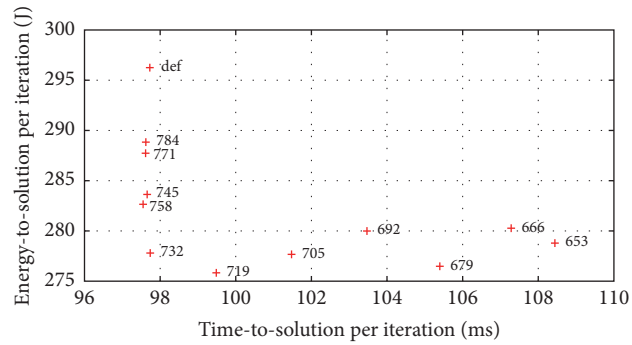

Figure 4: Energy consumption of a single node of the COKA cluster for 10 k iterations of a D2Q37 Lattice-Boltzmann simulation, over 16384 × 8192 lattice points, using the 16 available GPUs. Energy consumption is reported in Joule per iteration for the different possible GPU clock frequencies.

5.2. LHCb Software. LHCb [38] is one of the four main experiments collecting data at the LHC (Large Hadron Collider), the world's largest particle accelerator located at CERN laboratory in Geneva. Its purpose is to investigate b-hadron decays with high statistics and precision, aiming mainly at the study of observables and rare decays violating CP (C, charge conjugation, and P, parity symmetries). The combination of C- and P-symmetries is found to be violated in several decay processes and among the others in the b-quark hadron decays. Studying such decays will give precise measurements of CP violating processes observables, which furthermore may depend on new physics effects.

The computing model of LHCb requires the reconstruction and analysis of simulated and real data. The software stack consists of a group of packages for the generation of simulated events and their reconstruction and analysis. Testing the full software stack on SoC architecture can provide valuable hints for the evolution of the computing model towards the third period of LHC data taking, due to start

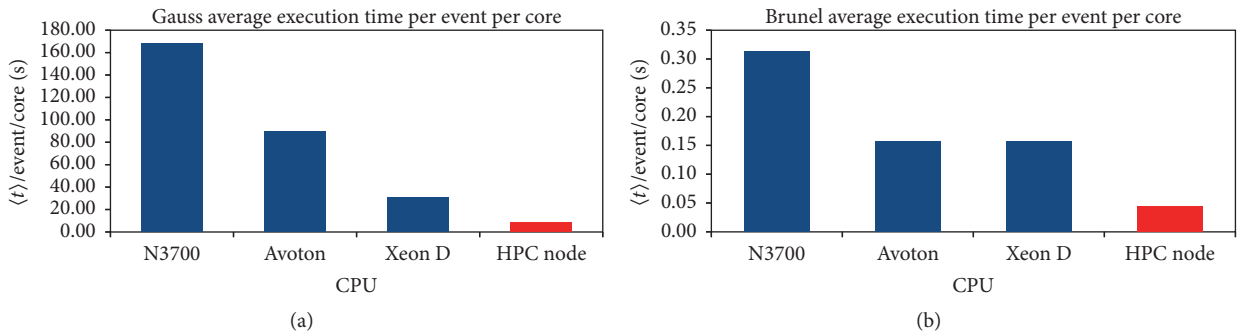

Figure 5: CPU time per event for the x86 architectures tested. (a) shows the results for the Gauss application, while (b) shows the Brunel application.

in 2021. Simulating data is usually a CPU-bound task, given the need to generate the events produced by the colliding beams, the interaction with the detector of all the particles, and the response of the readout electronics. In general, this task is not IO-bound, whereas the data reconstruction, either of real or simulated data, requires the access of data which are geographically distributed in several storage areas.

In order to determine the performances of the various available processors, we decided to set up a simulation task using the LHCb application Gauss [39] and a reconstruction task using the software package Brunel [40], which takes the raw detector data and builds physics objects starting from kinematic quantities like tracks, vertexes, and particle identification quantities. The CPU and IO requirements of the two applications are in general different. With the evaluation of the CPU performances and power consumption being the purpose of these tests, we decided for the reconstruction task to download the input file in advance. Furthermore, since the LHCb software stack is compiled for x86 architectures, we chose to run the tests on the following CPUs: Intel Pentium N3700, Intel Avoton C2750, Intel Xeon D-1540, and Dual Intel Xeon E5-2683 v3.

In Figure 5, the average timing per event of the four CPUs is reported. The histograms in (a) refer to the Gauss tests, while the ones in (b) refer to the Brunel tests. Simulation tasks usually take a small input file and produce a much bigger one (~1 GB), whereas reconstruction tasks elaborate the information of an input file with an event-based data structure. So the latter application reads the file at each iteration, hence requiring frequent access to the memory. As is clear from Figure 5, the execution times for the simulation task are strikingly different, reflecting the low profile of the Intel Pentium N3700 with respect to the other architectures. In particular, the CNAF HPC node is around 20 times faster than N3700 for the Gauss simulation task (8 s versus 168 s) and around 8 times faster (4 s versus 31 s) than N3700 for the reconstruction task (Brunel). The discrepancy in execution time for the reconstruction task is mitigated by the memory access. Thus, for the application that requires memory access, a low TDP solution could be profitably considered.

Then, Figure 6 shows the effects of taking into account power consumption in order to compare architectures. The metric shown in the latter figure includes both the execution time and the absorbed power (since energy is the product of the two), which has been measured using the *multimeter* system described in Section 3.

Taking the energy per event as a reference metric, the N3700 SoC is around 4.5 times more efficient than the CNAF HPC node for the simulation (Gauss) task (2.1 J versus 9.4 J) and around 13.5 times more efficient than the CNAF HPC node for the reconstruction (Brunel) application (0.002 J versus 0.027 J), confirming and actually reinforcing our statement above; that is, low-power CPUs such as the N3700 are the best performing in case of not-negligible IO applications.

Of course, it should be noted that many of these CPUs would be required in order to provide the throughput needed by LHCb, and we are planning to design a compact system consisting of low-power CPUs and providing a throughput comparable to a usual high-end server.

5.3. Neural Networks. Spiking neural networks play a dual role, depending on the scale of the simulated models: they contribute to a scientific grand-challenge, that is, the understanding of brain activity and of its computational mechanisms, and, by inclusion in embedded systems, they enhance the ability of applications like autonomous navigation, surveillance, and robotics. Therefore, fast and power-efficient execution of spiking neural network models assumes a driving role, at the cross-road between embedded and high-performance computing, shaping the evolution of the architecture of specialized and general-purpose multicore/many-core systems. See, for example, the TrueNorth [41] low-power specialized hardware architecture for embedded applications and [42] about the power consumption of the SpiNNaker specialized hardware architecture, based on the combination of embedded multicores and a dedicated networking infrastructure. About the strategy based on more standard HPC platforms and general-purpose simulators, see, for example, [43, 44].

Indeed, the quantitative codesign of the EURETILE many-tile execution platform and its many-process programming environment [45] motivated INFN APE lab to start the development of a Distributed Simulator of Spiking Neural Network with Spike-Timing Dependent Synaptic Plasticity (DPSNN-STDP) [46] mini-application benchmark.

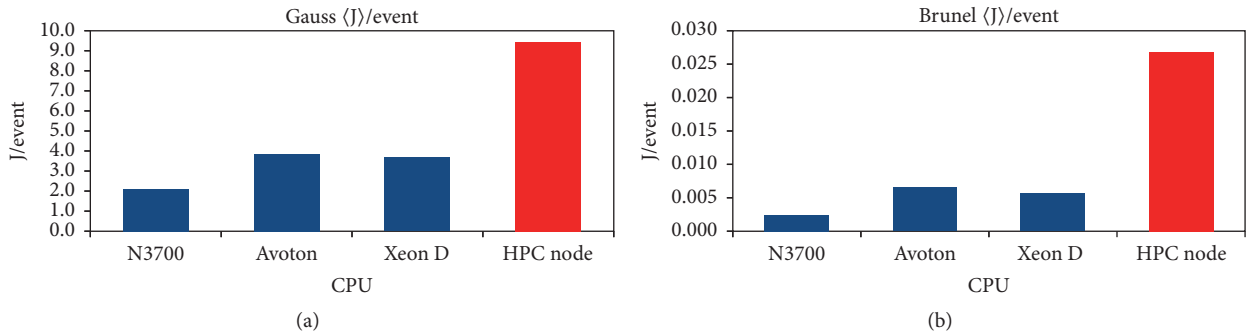

Figure 6: Energy consumption per event for the x86 architectures tested. (a) shows the results for the Gauss application, while (b) shows the Brunel application.

Currently, the WaveScalES experiment in the Human Brain Project uses this engine to simulate the activity of cortical slow waves on high-resolution models of cortical areas, including several tens of billions of synapses and the ExaNeSt [15] project includes DPSNN in the set of benchmarks that drive the development of future interconnects for platforms including millions of embedded ARM cores.

In [47], a first comparison of the power, energy, and speed of execution on ARM cores versus Intel Xeon quad-cores is reported. There, DPSNN has been run on NVIDIA Jetson TK1 boards (which include a quad-core ARM Cortex-A15 @2.3 GHz, 28 nm CMOS technology) and on clusters mounting quad-core Intel Xeon CPU (E5-620 @2.4 GHz, 32 nm CMOS). Here we extend the measures to the new-generation NVIDIA Jetson TX1 SoC based on ARMv8 architecture. Jetson TX1 includes four ARM Cortex-A57 cores plus four ARM Cortex-A53 cores in big.LITTLE configuration. We have measured its performances in executing the DPSNN code along with those of a coeval mainstream Intel processor architecture using a hardware/software configuration suitable to extrapolate a direct comparison of time-to-solution and energy-to-solution at the level of the single core. The energy consumption has been measured using the *multimeter* system described in Section 3.

We have used a Jetson TX1 board and a Supermicro SuperServer 7048GR-TR with two hexa core Intel E5-2620 v3 @ 2.40 GHz as hardware platforms, and we have run four MPI processes on both, simulating 3 s of the dynamics of a network made of 10^4 Leaky Integrate and Fire with Calcium Adaptation (LIFCA) neurons connected via 18×10^6 synapses. Results are shown in Figures 7 and 8 and can be summarized as follows: although the x86 core architecture is about five times faster than the ARM Cortex-A57 core in executing the simulation, the energy it consumes to do it is about three times higher than the energy consumed by the ARM Cortex-A57 core.

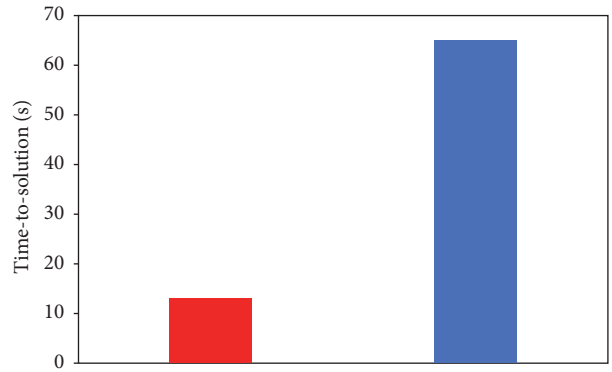

Figure 7: Time-to-solution for the DPSNN simulation on the x86 Haswell (left) and Jetson TX1 (right).

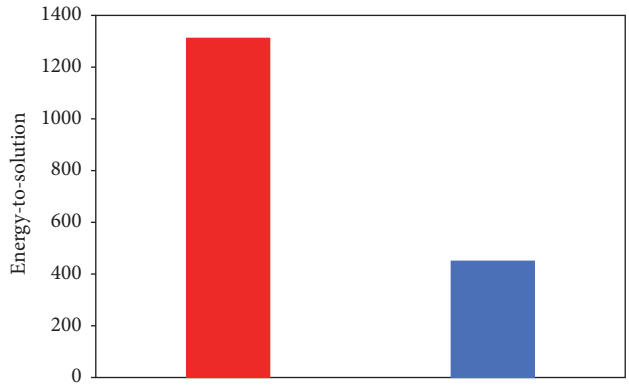

Figure 8: Energy-to-solution (in Joule) for the DPSNN simulation on the x86 Haswell (left) and Jetson TX1 (right).

5.4. Computed Tomography. Among the various scientific applications explored within the COSA project, X-Ray Computed Tomography turned out to be a particularly well suited case for the use of low-power SoCs, as described in [48].

X-ray Computed Tomography can be gainfully applied to the field of Cultural Heritage in order to reconstruct the internal structure of art objects in a noninvasive way for both scientific investigations and restoration purposes. This is typically both time-consuming and power-consuming; also, in most situations, executing the reconstruction software directly where and when the X-ray measurements are acquired is infeasible. Hence, the possibility of running the reconstruction algorithm on a mobile, possibly battery-powered, device is particularly appealing.

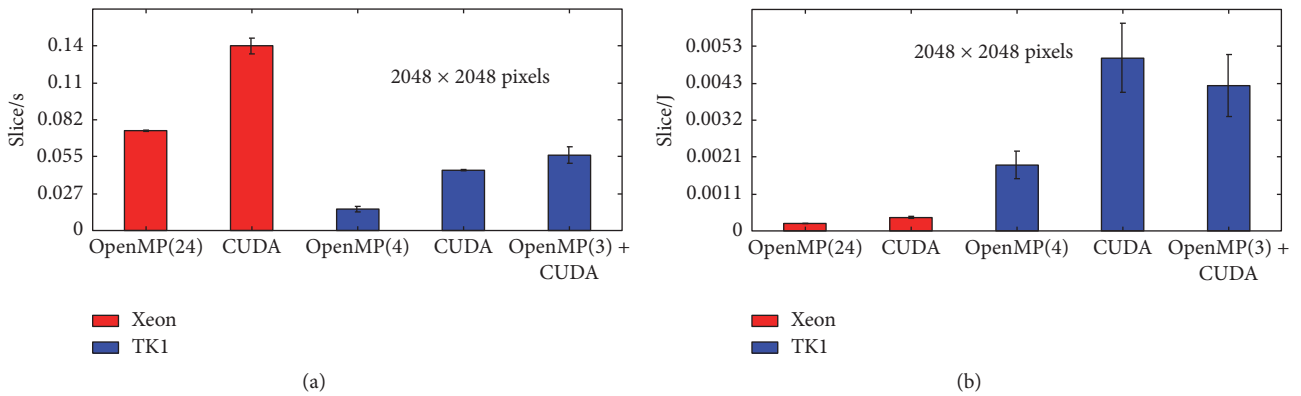

Figure 9: Slices per second (a) and per Joule (b) of OpenMP and CUDA versions of the Filtered Backprojection algorithm executed on traditional high-end Xeon architecture (red) and on Jetson TK1 board (blue). The rightmost bar shows the combined GPU + CPU solution on Jetson TK1; that is, only 3 CPU threads are used for the OpenMP version. Otherwise, the OpenMP version uses all the available threads, that is, 24 threads on Xeon and 4 threads on Jetson TK1.

For our study, we have considered the C and MPI Filtered Backprojection algorithm for Computed Tomography reconstruction [49] developed by the X-ray Imaging Group of the Physics and Astronomy Department at the University of Bologna [50]. The Filtered Backprojection algorithm is heavily used in 3D Tomography reconstruction, and performances can be easily measured in terms of 2D slices (of a 3D volume) reconstructed per time unit and per energy unit.

We have exploited the Graphics Processing Unit (GPU) of the Jetson TK1 SoC, available in the COSA cluster located at CNAF, and maximized the simultaneous use of CPU and GPU by combining a multithreaded OpenMP version and a GPU-CUDA version of the reconstruction algorithm, which have been executed in parallel on the SoC. We ran the OpenMP and CUDA implementation in two different SSH sessions at the same time on the same platform. We have used three OpenMP threads for the CPU implementation, running on three of the four available cores, and one thread for the CUDA implementation, running on the fourth core.

Figure 9 compares the numbers of slices per time unit (a) and per energy unit (b) reconstructed using the CPU (OpenMP version) and the GPU (CUDA version) on a Jetson TK1 SoC and on the reference Xeon server equipped with a NVIDIA K20 GPU for a characteristic image from the considered dataset (we refer the reader to [48] for more details). Power consumption has been measured using the *multimeter* system described in Section 3.

In [48], we showed that only three Jetson K1 boards equipped with Giga Ethernet interconnections allow reconstructing as many 2D slices (of a 3D volume) per unit time as a traditional high-performance computing node, using one order of magnitude less energy.

It is important to note that the reconstructed images have been always compared in terms of pixel-by-pixel standard deviations with the image reconstructed using the original, serial code to check the correctness of the reconstruction process. Therefore, our results seem to be very promising in view of the construction of an energy-efficient computing system of a mobile tomographic apparatus.

5.5. Einstein Toolkit. The scientific problem considered in this application is a high-resolution simulation of inspiral and merger phase of binary neutron star systems, which are among the most likely sources of gravitational waves targeted for detection by LIGO/Virgo [51–53]. The numerical setup of the test is based on the Einstein Toolkit (ET) Consortium code [54], which performs the time evolution of matter coupled to Einstein's equations (General Relativistic Hydrodynamics), and it is the same as that used in production setting and described in detail in [55].

The numerical complexity of the code reflects the need to compile the whole Einstein Toolkit [56], which is an open set of over 100 components (Cactus thorns) for computational relativity and consists of 1000 source files written in C, C++, F77, and F90 that implement OpenMP parallelism and MPI distribution of execution and partition of workloads and memory allocations. In this setting, the following are included:

(i) Cactus framework for parallel high-performance computing

(ii) Mesh refinement with Carpet

(iii) Matter evolution with GRHydro [57]

(iv) Metric evolution using McLachlan BSSN evolution of the matrix

(v) Initial data computed using the LORENE CODE.

Simulations based on this framework are routinely executed on all the major HPC systems and the basic performance test reported here refers to the "Galileo" Tier-1 HPC system. Galileo is located at the CINECA HPC center in Bologna [58] and is partially financed by INFN. On this system, we performed scaling reference test (see Figure 10) using different resolutions, ranging from 0.75 to 0.09375 (corresponding to 1100 m to 138 m). In order to perform a simulation of 30 ms of physical time, using resolution $dx = 0.25$ (the finest grid), a production run on Galileo requires a week on 256 cores and allocates 108 GB of physical RAM on the system.

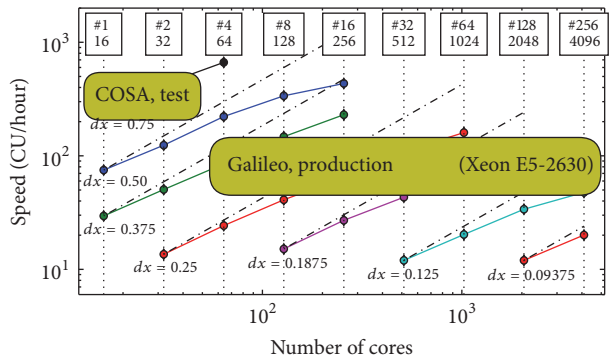

Figure 10: Galileo system at CINECA (1 MPI process per socket).

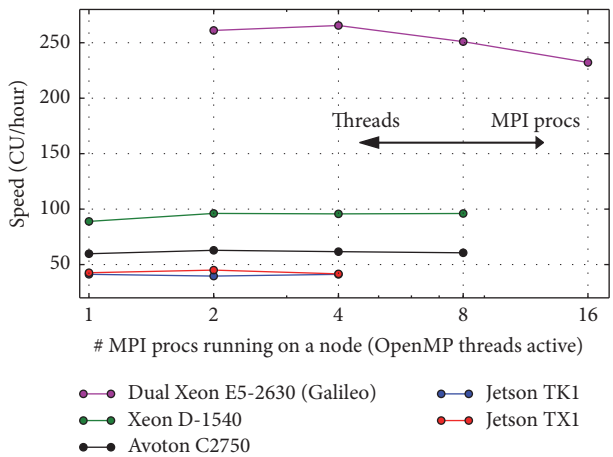

Figure 11: MPI processes versus OpenMP threads performance on a single whole node.

The code has been successfully compiled and run on several COSA boards. The ET "Hilbert" (ET_2015_05) stable release has been compiled on all the four considered COSA platforms and the binary codes have been executed using parallel distribution with the MPI and OpenMP programming paradigms. This is very good news, as it shows that the software environment available on the considered architectures is enabled to run actual HPC applications.

The second objective has been the evaluation of the performance that the SoCs are able to deliver on this real simulation case. Due to the memory constraint of the COSA platforms, we had to impose additional symmetry and a very coarse resolution ($dx = 0.75$) in order to limit memory configuration below 2 GB. We evolved the system on a configuration with cubic multi-Grid mesh with five levels of refinement and we performed 800 time evolution steps.

In Figure 11, the performances of the tested systems are shown on a single node in order to avoid possible effects of the communication network in place. All the available cores have been allocated with only MPI processes or only OpenMP threads or mixed processes and threads. This gives an assessment of the CPU speed and intranode performance, which are consistent with respect to the relationship between the peak performances of the different cores.

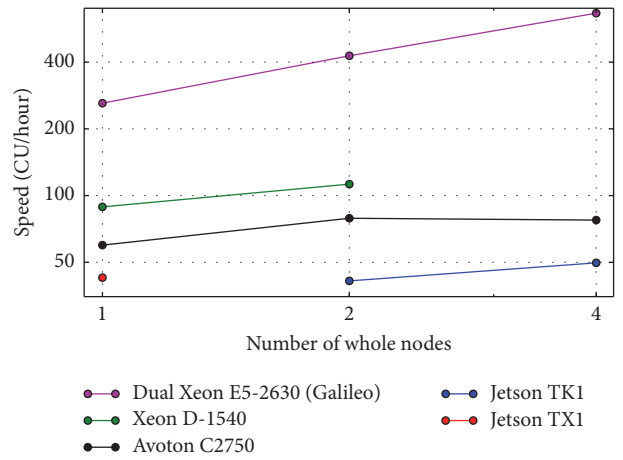

Figure 12: Simulation on Galileo and COSA architectures, with resolution $dx = 0.75$.

The internode communication has been stressed by scaling from 1 to the maximum number of available nodes (see Figure 12). This procedure shows the possibility to scale and distribute the workloads between different nodes, but since the communication in the COSA CNAF cluster is based on Ethernet technology instead of high-speed interconnection technology (like InfiniBand on Galileo), the plots show poor scalability and confirm that high-speed communication between nodes should be adopted in order to use such system for real HPC applications.

5.6. Bioinformatics. We have also tried to run on the COSA CNAF cluster a few significant applications from the Bioinformatics realm, as described in [59]. In the tests described below, power consumption has been measured using the *multimeter* system described in Section 3.

First, we have considered GROMACS (GROningen MAchine for Chemical Simulations, [60]), free, open-source software for molecular dynamics simulations used worldwide to simulate the Newtonian equations of motion for systems with hundreds to millions of particles, for example, proteins or polymers. GROMACS is written in ANSI C, and both a parallel version using standard MPI and a GPU accelerated version using CUDA are available.

The porting of GROMACS version 5.1 to Jetson TK1 platform has been easy, and simulations of a real-life use case (approximately 55000 atoms) have been run using 32 cores, that is, 8 Jetson TK1 boards connected through 1 Gbit/s Ethernet and CPU + GPU on a single node. In the CPU-only multicore mode, the SoC was proven to be ten times slower than the reference high-end server equipped with two Intel Xeon E5-2620 v2 CPUs. However, it is slightly better in terms of power ratio. Instead, for the CPU + GPU run on just one node, the Jetson TK1 is only 5.5 times slower but 6.6 times more power-efficient.

Furthermore, we ported a space-aware system biology stochastic simulator to the newer Jetson TX1 boards. The considered simulator couples Dynamical Probabilistic membrane systems [61] with a modified version of the τ-leaping

stochastic simulation method [62, 63], which takes into account size and volumes of objects in crowded systems and allows modeling biochemical systems executing thousands of simulations in parallel.

Again, the porting of the Sτ-DPP algorithm to low-power Jetson TX1 boards has been straightforward, and computing performances are satisfying (2-3 times slower than the reference high-end Xeon servers), with 10 times lower power consumption. As is the case with all the considered applications, performances become poor when attempting real multinode runs due to the high network latency of the interconnection.

6. Conclusions

In this paper, we have presented the results coming from the COSA (Computing On SOC Architecture) project, aiming at investigating novel hardware platforms and software techniques that can be exploited to realize energy-efficient computing systems for scientific applications. The hardware platforms taken into consideration are those with high ratio of flop per watt. We have considered standard multicore and many-core devices such as CPUs and GPUs and *System on Chip* (SoC) developed for the mobile and embedded realms. These sectors and the HPC world have historically been very isolated, but we are now experiencing very important convergence between the two markets. This is true for what concerns constraints (i.e., energy-efficiency) and needs (i.e., computing power). The COSA project is investigating the possibility to exploit this convergence to increase performance/power ratio in computing systems running scientific applications. Porting real-life applications from various scientific fields to low-power Systems on Chip architectures derived from the embedded and mobile markets is one of the main objectives of the project. Hardware platforms based on both Intel and ARM architectures have been considered and benchmarked with synthetic tests and real-life applications belonging to various scientific fields, ranging from High Energy Physics to Biomedicine, Theoretical Physics, Computer Tomography, and Neural Network simulations.

The results of our work show that it is now actually possible, and in some cases even very easy, to compile and run complex scientific workloads on low-power, off-the-shelf devices not conceived by design for high-performance scientific applications.

For certain applications, the computing performances are satisfying and even comparable to those obtained on traditional high-end servers, with much lower power consumption, in particular if the GPU available on SoCs is exploited.

Comparing high-end HPC servers with development boards hosting low-power Systems on Chip taken from the mobile and embedded world is clearly *not fair* if only energy aspects are taken into account. In fact, there are workloads that simply cannot be run on off-the-shelf hardware and embedded platforms, but, nevertheless, the work presented in this paper showed that, for certain applications, even parallel applications, the use of low-power architectures represents a feasible choice in terms of tradeoff between execution times and power drain. However, the high-end platforms are still capable of better absolute computing performance. Among the limitations that low-power SoCs-based platforms still show, we can list the following: small maximum RAM size, high latency of the network connections, few and small PCI slots, and missing support for ECC memory. For all these reasons, off-the-shelf systems can hardly be used for extreme scale applications and highly demanding HPC computations; anyhow, our experience shows that systems based on low-power SoCs can be a viable solution to reduce power consumption if a proper integration is carried on. Indeed, the exploitation of this kind of hardware in a production environment requires by definition many nodes to be integrated, and the costs of such integration should be carefully analyzed and taken into account when making comparisons with standard architectures systems. The analysis of this integration cost is out of the scope of this paper and is demanded to a future work.

In general, performances become poor when attempting real multinode runs on platforms without native Ethernet connections. Some of these platforms, in fact, implement the network connections through slow USB-Ethernet bridges. However, almost all the SoCs under evaluation provide PCI lanes that, if supported by motherboards with proper connectors, can be exploited to install low-latency cards allowing for network performances comparable to those obtained in standard high-end servers. However, to further improve the network performance of SoCs-based system, the project is also investigating the adoption of custom toroidal interconnections prototyped with state-of-the-art FPGAs.

As for high-end platforms, whenever the scientific workloads succeed in exploiting the large number of graphics cores available in the SoCs, it is possible to obtain good speed gains and increase in computing power per watt. This is true also for the considered applications and in particular for those that we have been able to run on the NVIDIA Jetson platforms (both TX1 and TK1) exploiting the CUDA environment.

Another research line that the project is investigating to reduce energy consumption of HPC application runs does not focus on hardware but on software. Modern operating systems, in fact, allow controlling the performance of various hardware components (i.e., tuning the RAM bus and CPU and GPU clocks) with libraries calls and APIs. Our results show that, if properly instrumented to reduce the GPU frequency when executing certain functions, the considered application can save $\approx$7% of the total consumed energy without impacting performances.

In the future, the COSA project will continue to analyze real-life applications performances on novel architectures, from both low-power and high-end ecosystems, including, for example, the recently released Intel Knights Landing many-core systems. In particular, for the low-power platforms, the performances of clusters with low-latency interconnections will be explored.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work has been jointly funded by the Scientific Commission V of the Italian Institute for Nuclear Physics (INFN) through the COSA project and by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement no. 671553 (ExaNeSt project) and Special Grant Agreement no. 720270 (the Human Brain Project SGA1, WP3.2 WaveScalES experiment). The authors would like to acknowledge the University of Ferrara and INFN for the access to the COKA cluster.

References

[1] [Online]. Available: http://www.cosa-project.it/.

[2] J. Choi, M. Dukhan, X. Liu, and R. Vuduc, "Algorithmic time, energy, and power on candidate HPC compute building blocks," in *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2014*, pp. 447–457, May 2014.

[3] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC?" in *Proceedings of the 2013 International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2013*, November 2013.

[4] The., "Mont-Blanc prototype: An alternative approach for HPC systems," in *Proceedings of the in SC16: International Conference for High Performance Computing, Networking*, pp. 444–455, 2016.

[5] [Online]. Available: http://www.montblanc-project.eu/.

[6] M. Marazakis et al., "EUROSERVER: share-anything scale-out micro-server design," in *Proceedings of the in 2016 Design, Automation & Test in Europe Conference & Exhibition*, pp. 678–683, Dresden, Germany, 2016.

[7] [Online]. Available: http://www.euroserver-project.eu/.

[8] J. P. Silva, E. Dufrechou, P. Ezzatti, E. S. Quintana-Ortí, P. Benner, and A. Remón, "Solving dense linear systems with hybrid ARM+GPU platforms," in *Proceedings of the 41st Latin American Computing Conference, CLEI 2015*, October 2015.

[9] V. P. Nikolskiy, V. V. Stegailov, and V. S. Vecher, "Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics," in *Proceedings of the 14th International Conference on High Performance Computing and Simulation, HPCS 2016*, pp. 682–689, July 2016.

[10] M. Geveler, D. Ribbrock, D. Donner et al., "The ICARUS White Paper: A Scalable, Energy-Efficient, Solar-Powered HPC Center Based on Low Power GPUs," in *Euro-Par 2016: Parallel Processing Workshops*, vol. 10104 of *Lecture Notes in Computer Science*, pp. 737–749, Springer International Publishing, 2017.

[11] D. Horak, L. Riha, R. Sojka, J. Kruzik, and M. Beseda, "Energy consumption optimization of the Total-FETI solver and BLAS routines by changing the CPU frequency," in *Proceedings of the 14th International Conference on High Performance Computing and Simulation, HPCS 2016*, pp. 1031-1032, July 2016.

[12] S. Catalán, J. R. Herrero, E. S. Quintana-Ortí, and R. Rodríguez-Sánchez, "Energy balance between voltage-frequency scaling and resilience for linear algebra routines on low-power multicore architectures," *Parallel Computing*, 2017.

[13] S. Furber and S. Temple, "Neural systems engineering," *Journal of the Royal Society Interface*, vol. 4, no. 13, pp. 193–206, 2007.

[14] [Online]. Available: http://apt.cs.manchester.ac.uk/projects/SpiNNaker/.

[15] M. Katevenis, N. Chrysos, M. Marazakis et al., "The ExaNeSt Project: Interconnects, Storage, and Packaging for Exascale Systems," in *Proceedings of the 19th Euromicro Conference on Digital System Design, DSD 2016*, pp. 60–67, September 2016.

[16] R. Ammendola, A. Biagioni, O. Frezza et al., "APEnet+: a 3D Torus network optimized for GPU-based HPC Systems," *Journal of Physics: Conference Series*, vol. 396, no. 4, p. 042059, 2012.

[17] R. Alfieri et al., "The COKA project," INFN-CNAF Annual report, 2014.

[18] [Online]. Available: http://www.fe.infn.it/coka.

[19] H. Jagode et al., "Power Management and Event Verification in PAPI," *Springer International Publishing*, pp. 41–51, 2016.

[20] [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[21] [Online]. Available: https://baltig.infn.it/COKA/PAPI-power-reader.

[22] E. Calore, S. F. Schifano, and R. Tripiccione, "Energy-performance tradeoffs for HPC applications on low power processors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9523, pp. 737–748, 2015.

[23] [Online]. Available: http://www.hpcg-benchmark.org/index.html.

[24] [Online]. Available: http://www.openblas.net/.

[25] S. Succi, *The Lattice-Boltzmann Equation*, Oxford university press, Oxford, UK, 2001.

[26] M. Sbragaglia, R. Benzi, L. Biferale, H. Chen, X. Shan, and S. Succi, "Lattice Boltzmann method with self-consistent thermo-hydrodynamic equilibria," *Journal of Fluid Mechanics*, vol. 628, pp. 299–309, 2009.

[27] A. Scagliarini, L. Biferale, M. Sbragaglia, K. Sugiyama, and F. Toschi, "Lattice Boltzmann methods for thermal flows: Continuum limit and applications to compressible Rayleigh-Taylor systems," *Physics of Fluids*, vol. 22, no. 5, Article ID 019004PHF, pp. 1–21, 2010.

[28] M. Pivanti, F. Mantovani, S. F. Schifano, R. Tripiccione, and L. Zenesini, "An Optimized Lattice Boltzmann Code for BlueGene/Q," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, Ed., vol. 8385 of *Lecture Notes in Computer Science*, pp. 385–394, Springer, Berlin, Germany, 2014.

[29] F. Mantovani, "Performance issues on many-core processors: A D2Q37 Lattice Boltzmann scheme as a test-case," *Computers & Fluids*, vol. 88, pp. 743–752, 2013.

[30] L. Biferale, "An optimized D2Q37 lattice boltzmann code on GP-GPUs," *Computers & Fluids*, vol. 80, pp. 55–62, 2013.

[31] E. Calore, A. Gabbana, J. Kraus, E. Pellegrini, S. F. Schifano, and R. Tripiccione, "Massively parallel lattice-Boltzmann codes on large GPU clusters," *Parallel Computing. Systems & Applications*, vol. 58, pp. 1–24, 2016.

[32] E. Calore, A. Gabbana, J. Kraus, S. F. Schifano, and R. Tripiccione, "Performance and portability of accelerated lattice Boltzmann applications with OpenACC," *Concurrency Computation*, vol. 28, no. 12, pp. 3485–3502, 2016.

[33] G. Crimi, F. Mantovani, M. Pivanti, S. F. Schifano, and R. Tripiccione, "Early experience on porting and running a Lattice Boltzmann code on the Xeon-Phi co-processor," *Procedia Computer Science*, vol. 18, pp. 551–560, 2013.

[34] E. Calore, N. Demo, S. F. Schifano, and R. Tripiccione, "Experience on Vectorizing Lattice Boltzmann Kernels for Multi- and Many-Core Architectures," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, Ed., vol. 9573 of *Lecture Notes in Computer Science*, pp. 53–62, Springer International Publishing, 2016.

[35] L. Biferale, F. Mantovani, M. Sbragaglia, A. Scagliarini, F. Toschi, and R. Tripiccione, "Second-order closure in stratified turbulence: Simulations and modeling of bulk and entrainment regions," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 84, no. 1, Article ID 016305, 2011.

[36] L. Biferale, F. Mantovani, M. Sbragaglia et al., "Reactive Rayleigh-Taylor systems: Front propagation and non-stationarity," *EPL*, vol. 94, no. 5, p. 54004, 2011.

[37] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications," *Concurrency Computation*, 2017.

[38] A. Alves Jr., "The LHCb Detector at the LHC," *Journal of Instrumentation*, vol. 3, pp. 10–1088, 2008.

[39] M. Clemencic, G. Corti, S. Easo et al., "The LHCb simulation application, gauss: Design, evolution and experience," *Journal of Physics: Conference Series*, vol. 331, no. 3, Article ID 32023, 2011.

[40] [Online]. Available: http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/brunel/.

[41] P. A. Medolla, J. V. Arthur, and R. Alvarez-Icaza, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[42] E. Stromatias, F. Galluppi, C. Patterson, and S. Furber, "Power analysis of large-scale, real-time neural networks on SpiNNaker," in *Proceedings of the 2013 International Joint Conference on Neural Networks, IJCNN 2013*, August 2013.

[43] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Communications of the ACM*, vol. 54, no. 8, Article ID 1978559, pp. 62–71, 2011.

[44] M.-O. Gewaltig and M. Diesmann, "NEST (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[45] P. S. Paolucci, A. Biagioni, L. G. Murillo et al., "Dynamic many-process applications on many-tile embedded systems and HPC clusters: The EURETILE programming environment and execution platforms," *Journal of Systems Architecture*, vol. 69, pp. 29–53, 2016.

[46] P. S. Paolucci, R. Ammendola, A. Biagioni et al., "Distributed simulation of polychronous and plastic spiking neural networks: strong and weak scaling of a representative mini-application benchmark executed on a small-scale commodity cluster," https://arxiv.org/abs/1310.8478.

[47] "Power, Energy and Speed of Embedded and Server Multi-Cores applied to Distributed Simulation of Spiking Neural Networks: ARM in NVIDIA Tegra vs Intel Xeon quad-cores," https://arxiv.org/abs/1505.03015.

[48] E. Corni, L. Morganti, M. P. Morigi et al., "X-Ray Computed Tomography Applied to Objects of Cultural Heritage: Porting and Testing the Filtered Back-Projection Reconstruction Algorithm on Low Power Systems-on-Chip," in *Proceedings of the 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016*, pp. 369–372, February 2016.

[49] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*, IEEE Service Center, Piscataway, NJ, USA, 1988.

[50] R. Brancaccio, M. Bettuzzi, F. Casali et al., "Real-time reconstruction for 3-D CT applied to large objects of cultural heritage," *IEEE Transactions on Nuclear Science*, vol. 58, no. 4, pp. 1864–1871, 2011.

[51] [Online]. Available: https://www.ligo.caltech.edu/.

[52] [Online]. Available: http://www.ego-gw.it/public/about/whatis.aspx.

[53] B. P. Abbott et al., "Observation of gravitational waves from a binary black hole merger," *Physical Review Letters*, vol. 116, Article ID 061102, 2016.

[54] The Einstein Toolkit. [Online], http://www.einsteintoolkit.org.

[55] R. De Pietri, A. Feo, F. Maione, and F. Löffler, "Modeling equal and unequal mass binary neutron star mergers using public codes," *Physical Review D*, vol. 93, no. 6, Article ID 064047, 2015.

[56] F. Löffler, J. Faber, E. Bentivegna et al., "The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics," *Classical and Quantum Gravity*, vol. 29, no. 11, p. 115001, 2012.

[57] P. Mösta, B. C. Mundim, J. A. Faber et al., "GRHydro: a new open-source general-relativistic magnetohydrodynamics code for the Einstein toolkit," *Classical and Quantum Gravity*, vol. 31, no. 1, p. 015005, 2014.

[58] [Online]. Available: https://www.cineca.it/en.

[59] L. Morganti, D. Cesini, and A. Ferraro, "Evaluating Systems on Chip through HPC Bioinformatic and Astrophysic Applications," in *Proceedings of the 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016*, pp. 541–544, February 2016.

[60] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen, "GROMACS: fast, flexible, and free," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1701–1718, 2005.

[61] E. Mosca, P. Cazzaniga, D. Pescini, G. Mauri, and L. Milanesi, "Modelling spatial heterogeneity and macromolecular crowding with membrane systems," in *Membrane Computing*, vol. 6501 of *Lecture Notes in Computer Science*, pp. 285–304, Springer, Berlin, Germany, 2011.

[62] P. Cazzaniga, D. Pescini, D. Besozzi, and G. Mauri, "Tau leaping stochastic simulation method in P systems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4361, pp. 298–313, 2006.

[63] Y. Cao, D. T. Gillespie, and L. R. Petzold, "Efficient step size selection for the tauleaping simulation method," *The Journal of Chemical Physics*, vol. 124, no. 4, Article ID 044109, 2006.