The Investigation of the ARMv7 and Intel Haswell Architectures Suitability for Performance and Energy-Aware Computing

Vojtech Nikl, Michal Hradecky, Jakub Keleceni, and Jiri Jaros

IT4Innovations Centre of Excellence
Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 61200 Brno, Czech Republic,
{inikl,jarosjir}@fit.vutbr.cz
hradec.m@gmail.com
jakub.keleceni@gmail.com

Abstract. The reduction of the CPU frequency and voltage is a well-known approach to improve energy consumption of memory-bound applications. This is based on the conception that the performance of the main memory sees little or no degradation at reduced processor clock speeds while power consumption decreases significantly improving the overall energy efficiency. We study this effect on the Haswell generation of Intel Xeon processors as well as the ARMv7 generation of the 32-bit ARM big.LITTLE architecture. The goal is to analyse and compare computational performance, energy consumption and energy efficiency on a series of tasks, each focusing on different parts of the system and provide an analysis and generalisation to other similar architectures.
The benchmark suit consists of compute and memory intensive benchmarks as well as both single and multi-threaded scientific applications. The results show that frequency and voltage scaling can significantly improve algorithms' energy efficiency. Up to 2.5× on ARM and 1.5× on Intel compared to the maximum frequency. ARM is up to 2× more efficient than Intel.

Keywords: Haswell, ARMv7, Odroid XU4, k-Wave, LAMMPS, energy efficiency.

1 Introduction

Nowadays, the energy efficiency of modem processors is becoming more and more important next to the overall performance itself. Many programming tasks and problems do cannot use the hardware very efficiently due to being memory or communication-bound. Many clock cycles are wasted while waiting for data or a dependency conflict. Therefore, it is often not beneficial to use faster chips to achieve better runtimes. In this case, underclocking and undervolting, or employing slower low power processors or accelerators may be much more efficient. Mainly because of the possibility to get the same results using much less energy and often without any significant performance penalties.

An average Intel Xeon processor provides around 150–300 GFlop/s in double precision, with the Thermal Power Design (TDP) of 85–130 W. This gives roughly 2 GFlops/W of peak energy efficiency. These chips consist of about 6–18 cores being the most widely used CPUs in today's high performance clusters and supercomputers, according to the Top500[1] ladder.

Searching for even better efficiency, mobile ARM processors have attracted a lot of interest since their performance is comparable to the x86 CPUs. For example, an ARM based development board Nvidia Tegra X1[2] and its GPU can provide 512 GFlop/s while consuming only about 11 W of energy. This yields almost 50 GFlops/W in single precision.

The Green500[3] list provides a ranking of the most efficient supercomputers in the world. The most efficient machine reaches almost 10 GFlops/W using the nVidia DGX-1 system[4]. Current estimates indicate that processor efficiency will have to evolve to 50 GFlops/W for exascale machines to meet the realistic power budget of 20 MW.

Last but not least, an important reason to focus more on power efficiency is the resource allocation policy of supercomputing centers. Currently, resources are distributed among users based on core-hours. The energy efficiency of users' applications is defined simply by the runtime, faster is almost always more efficient. The processors are almost always running at the highest possible frequency, even when the application being executed may not fully utilise the processors' resources. This leads to a lot of wasted energy. However, due to rapidly increasing energy demands of modern clusters, the way the resources are allocated may change. Instead of using the core-hour metric, the users will be charged based on consumed kWhs. Hand to hand with this approach, users will be able to manually change hardware parameters such as frequency and voltage or shut down parts of the system. The SuperMUC[5] supercomputer already provides a frequency scaling options in the job scheduler for its users. The Taurus[6] supercomputer additionally allows users to change the processor frequency dynamically during job runtime. Taurus is able to measure the energy consumption of each of its Haswell nodes using a built-in FPGA probe. This is going to put much more emphasis on energy efficiency from both the software and hardware viewpoints. The hardware side will be much more dynamic and the ways to use provided resources as efficiently as possible will have to be exploited and optimised.

Much research effort in the area of the energy efficient computing makes use of the Dynamic Voltage and Frequency Scaling (DVFS) to improve energy efficiency [1], [8], [12], [6]. The incentive is often that a system's main memory bandwidth is unaffected by reduced clock speeds, while the power consumption decreases significantly. The examples of memory-bound algorithms are sorting and searching algorithms, sparse vector-matrix algebra or multidimensional fast Fourier transforms, assuming the input data cannot fit into caches. These algorithms spend most of the time accessing the main

[1] https://www.top500.org/
[2] http://www.nvidia.com/object/tegra-x1-processor.html
[3] https://www.top500.org/green500/
[4] http://www.nvidia.com/object/deep-learning-system.html
[5] https://www.lrz.de/services/compute/supermuc/
[6] https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus

memory and their compute intensity is often very low. Besides DVFS, another approach might be to switch off unneeded cores to save energy and let the others work at maximum frequency. However, this requires a direct hardware support.

This paper presents a study of DVFS and its impact on the energy efficiency. The investigated architectures are the latest Haswell generation of x86_64 Xeon systems from Intel and ARMv7 big.LITTLE architectures. A series of different benchmarks is tested, ranging from synthetic compute and memory ones to scientific applications.

2 Related Work

The Mont-Blanc project [10] based in Barcelona, Spain, has developed a high performance parallel system based on the ARMv7 architecture and its Cortex-A15 cores. The system was compared to a production supercomputer MareNostrum III composed of the Intel Xeon Sandy Bridge architecture. A single node of Mont-Blanc is 9× slower while saving 40% energy. MPI applications are 3.5× slower using the same number of processes, but consuming 9% less energy. A single node of Mont-Blanc consumes 5.3 W and 9.5 W while idle or load, respectively. Very similar architectures are compared in our paper, however, instead of focusing on MPI and GPUs, the emphasis is put on single-threaded and multi-threaded applications.

The READEX project [11] is improving energy efficiency of applications in the field of High Performance Computing by means of dynamic auto-tuning. This allows users to automatically exploit the dynamic behaviour of their applications by adjusting the hardware parameters to match the actual resource requirements. Their software consists of 3 main parts, the Periscope Tuning Framework (PTF) for design time analysis, the READEX Runtime Library (RRL) for tuning at runtime and the Score-P framework for the instrumentation and measurements of HPC applications. The outcome of the automatic READEX methodology is expected to be at least 50% of the manually achievable gains. Similarly to READEX, the goal is to analyse and find the optimal settings for a specific system running a given algorithm or its kernel. READEX exploits dynamism during the application's runtime on the Intel x86_64 architecture only. In this paper, a static frequency is set for each run.

Choi et al. [3] conducted a microbenchmarking study of the time, energy and power consumption on several existing platforms including modern GPUs, ARM (Arndale dual-core Cortex-A15) and Intel processors (Nehalem and mobile Ivy-Bridge) and an Intel Phi KNC accelerator. The dual-core ARM Cortex-A15 achieved 2.2 GFlops/W and 0.56 GB/W, Intel Nehalem achieved 0.62 GFlops/W and 0.14 GB/W using an architecture-specific hand-tuned benchmark. Our paper focuses only on the ARM and Intel x86 architectures, however, it provides a much wider set of benchmarks using actual scientific HPC applications.

Huang et al. [7] analyse the energy consumption of both the compute (HPL) and memory-bound (STREAM) problems on Haswell E5-2600 v3 architecture. The PAPI RAPL framework [14] is used to track the energy consumption of different parts of the system. The effect of different P-states, hyper-threading, socket power imbalance and core affinity on power consumption and performance were analysed. The results showed that different P-state settings provide up to 33% energy savings. Enabling the hyper-threading and core affinity improves energy efficiency by 19–48%. Minor power imbalances can be observed between the two sockets. Compiler optimisations improved the energy demands by 28.6%. Regarding to this paper, three different P-states are analysed in terms of performance and energy efficiency, threads and processes are always pinned to the cores, maximum compiler optimisations are used and a wider range of benchmarks is tested.

Hackenberg et al. [5] analyse a number of Haswell energy features, such as the enhanced RAPL implementation with better accuracy, integrated voltage and frequency regulators for each core, lower and unpredictable clock frequency for workloads with substantial amounts of AVX instructions and the P-state (voltage and frequency operation point) transition behaviour. The most important information for this paper is that RAPL measurements were verified using several microbenchmarks avoiding interference effects due to time synchronisation. The results show almost perfect correlation to the total system power consumption (AC) measured with high-accuracy power meter.

The contribution of our paper is the comparison of two architectures on a unique set of benchmarks. Two different methodologies are used for expressing the energy efficiency. A unique hardware setup of the Samsung Odroid-XU4[7] kit, based on a more powerful power supply, cooling and an accumulative power consumption sensor, is utilised.

3 Investigated Systems

The system configurations, all the benchmarks were run on, are summarised in Table 1 and 2.

The operating system was Ubuntu 16.04 on both systems.

On Intel, the energy measurements were taken using the **Intel Performance Counter Monitor**[8] and its **pcm-power** module, which can directly access the Running Average Power Limit Model Specific Registers (RAPL MSRs) of the CPU. It measures the energy consumption of three main components of each CPU - *package*, *powerplane* and *dram*. The package measures the whole socket including the memory controller. Powerplane only measures the cores themselves and dram measures the corresponding DRAM modules. The powerplane measurements are not supported by the Haswell architecture. The total power consumption in Watts was calculated as

[7] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825
[8] https://software.intel.com/en-us/articles/intel-performance-counter-monitor

Table 1: Intel Haswell system hardware overview

Server	Supermicro 7048GR-TR	
Motherboard	Supermicro X10DRG-Q	
Processor	2× Intel Xeon E5-2620v3	TDP 2× 85 W, 2× 230.4 GFlop/s (SP, no Turbo), 2× 15 MB L3, 12× 256 KB L2, 12× 32 KB L1
RAM	2× 32 GB DDR4-2133	2× 59 GB/s, 2× 4 channels
Storage	SSD Crucial MX200 250 GB	

Table 2: Samsung ARMv7 system hardware overview

Device	Samsung Odroid-XU4	
Processor	Samsung Exynos5422 (4× Cortex-A15 + 4× Cortex-A7)	TDP ~15 W, 4×4+4×2.8 GFlop/s (SP) 2×2 MB L2, 8×32 KB L1
RAM	2 GB LPDDR3 933MHz	14.9 GB/s, dual channel
Storage	eMMC5.0 HS400 Flash Storage	

$$package0 + dram0 + package1 + dram1 \tag{1}$$

The Samsung Odroid-XU4 kit does not support any hardware counters for measuring power consumption. The energy consumption was measured by the **KCX-017**[9] USB meter connected in-between the power supply and the power connector of the board to display actual electric voltage and current. The current is also accumulated into mAh used to manually calculate the overall energy consumption. Each benchmark ran long enough or was run multiple times in a loop to consume at least 20 mAh. The average deviation caused by reading the display manually is about 5 %.

The original Odroid power supply is not sufficient during high loads, the voltage dropped below 4.2V and the kit got frozen. A programmable power supply **Diametral P230R51D**[10] was used instead.

The original cooler is also also sufficient during high loads. It was replaced by **Primecooler PC-NBHP1**, originally used for motherboards' north bridges. The heat-conducting tape was replaced by the Arctic Ceramique paste. This dramatically improved the temperatures, however, due to the plastic heat spreader, 95 °C was often reached and the processor began throttling under High Performance Linpack at 2000 MHz on 4× Cortex-A15. This resulted in slightly poorer results in this particular test. All other tests performed within the range of safe temperatures and did not alter the performance. The complete hardware setup is shown in Fig. 1.

Our Haswell CPU supports frequencies ranging from 1.2 to 2.4 GHz, excluding the Intel Turbo boost. To be able to manually set a chosen frequency, the Intel P-state driver had to be replaced with the ACPI driver and the governor (power scheme for the CPU) was set from *balanced* to *userspace* using the system's **cpupower** utility. Similarly on Odroid, the **cpufreq-set** utility was used to change the frequency.

All the benchmarks were compiled using the **GNU Compiler Collection 5.3.0** compiler. The optimisation flags used for Haswell and ARM respectively were

```
-O3 -mavx2 -mtune=native -march=native        (2)
```

```
-O3 -mfpu=neon-vfpv4 -mtune=cortex-a15 -march=armv7-a   (3)
```

[9] https://cdn.solarbotics.com/products/datasheets/kcx-017%20power%20bank%20testing.pdf

[10] http://diametral.cz/ac-dc-zdroje/dc-regulovatelne-zdroje/laboratorni/laboratorni-zdroj-p230r51d-2x-030v/4a-1x-5v/3a.html

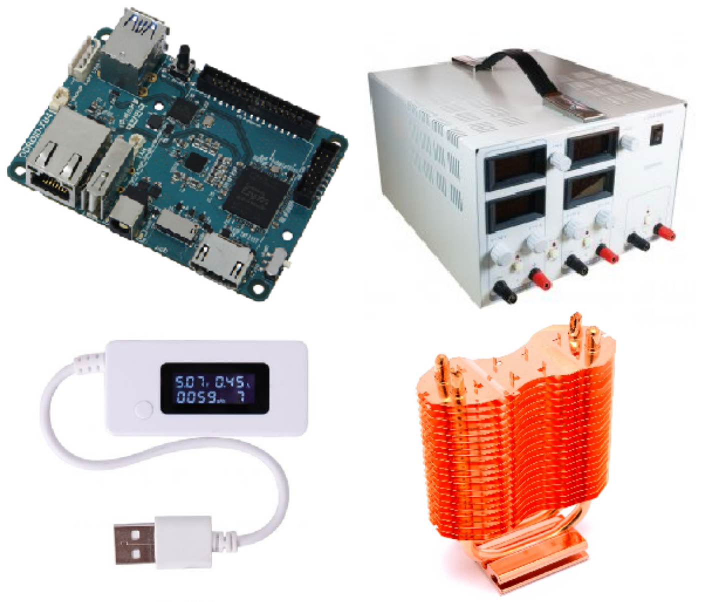

Fig. 1: A complete hardware setup for the Samsung Odroid-XU4 kit - the Diametral P230R51D power supply, the Primecooler PC-NBHP1 cooler and the KCX-017 power meter.

On Haswell, three main frequencies for all the cores were chosen to be benchmarked, 1.2, 1.8 and 2.4 GHz. The Haswell architecture supports setting an individual frequency for each core, however, this feature is not utilised because all the benchmarks were run on all 2×6 cores. The single-threaded ones where the uniform frequency was set to keep the results comparable. Intel Turbo boost was turned off. Voltages for all frequencies were set automatically based on the default CPU stepping provided by Intel. On Odroid, 200, 800 and 1400 MHz were chosen for both A7 and A15, in addition to 2000 MHz for A15. The Exynos processor supports switching off all the cores except the first A7 core. However, using even a single core from either the A7 or A15 quadcore cluster keeps the whole cluster running. Therefore switching off the A15 cluster only proved beneficial. All benchmarks run on the A7 cluster were executed with the A15 cluster cores switched off to further isolate the A7 cores in terms of energy demands.

One parallel benchmark was parallelised using **OpenMPI 1.8.4**, the rest using OpenMP. Each MPI process was bound to its core using the **mpirun** binding arguments, non-MPI and serial processes were bound using the **taskset** system utility. If threading was used, each thread was bound to its core using the `GOMP_CPU_AFFINITY` variable. All data arrays were aligned using `posix_memalign` to 64 bytes, which is the cache line width on both architectures' memory.

The number of floating point operations of each test run was obtained using the **PAPI** [14] library and the `PAPI_SP_OPS` event for single precision and the `PAPI_-DP_OPS` event for double precision floating point operations.

Two different metrics of understanding the energy demands are used in this paper called **the overall** and **the net**. The overall energy is the energy used by the whole system, in our case summing the RAPL readings from sockets and DRAMs on Haswell, or using the voltage and current readings from the USB meter on Odroid. The net energy is the overall energy minus the energy the system would require to run for the same amount of time in standby. This metric is marked ▼ in the following tables. For example, if the computation takes 10 s and the system's power consumption is 10 W under load and 2 W in idle, the overall energy demands are 100 Joules while the net energy demands are 80 Joules. This way, we can isolate the algorithm's energy requirements from the system's underlying overhead and also more accurately compare results across different architectures. Similar metrics are used in the READEX project [11].

4 Benchmarks

Three main groups of benchmarks were tested: synthetic ones focusing on CPU and memory, simple single-thread and parallel scientific applications.

The CPU's attainable performance was tested using the **High Performance Linpack 2.2** [4] compiled with the **ATLAS 3.10.3**[11] library. The compilation of ATLAS took over 24 hours on a single Odroid kit. The memory and cache subsystem was benchmarked using **LMBench 3** [9], which can measure read/write bandwidth and latency on data of a chosen size.

The single-threaded benchmarks consisted of the **Linpack** benchmark, a recursive **quicksort**, an iterative **calculation of π** using a continued fraction, and a recursive **Fibonacci series** calculations. The quicksort focuses mainly on random data access and can be considered a memory-bound problem for large input data. The π calculation is a representative of common naive codes written by a non-HPC user. The Fibonacci series is similarly naive and focuses on the stack usage and its implementation.

The multi-threaded algorithms consisted of **LAMMPS** [2], a molecular dynamics simulator and its Fene bead/spring benchmark (polymer melt system with 32 000 atoms), the **k-Wave** toolbox [13], an ultrasound simulation toolkit based on a k-space pseudospectral method, and a 2D heat propagation algorithm using the 4^{th} order Finite Difference Time Domain (FDTD) method in space and the 1^{st} order in time.

5 Experimental Results

This section presents the results measured on both the Intel Haswell and ARMv7 architectures. Colours in tables represent the order of the particular result in a given group of results (red being the worst, yellow being the median and green being the best, tables with rows separated by a small vertical space have rows coloured separately.

5.1 Synthetic CPU and Memory Benchmarks

While Haswell being the most powerful CPU in the HPL benchmark, it is also the most energy efficient chip in both the overall and net parameters (see Table 2). Generally, an

[11] http://math-atlas.sourceforge.net/

Table 2: Performance, overall and net energy efficiency for High Performance Linpack.

	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
GFlop/s	0.293	1.15	1.98	1.21	4.77	7.68	9.34	201	251	300
GFlops/W	0.111	0.316	0.458	0.563	0.923	0.895	0.532	1.37	1.54	1.47
GFlops/W▼	1.10	1.45	1.47	1.50	2.23	1.46	0.686	2.20	2.23	1.95

worst 🟥🟨🟩 best (each row coloured separately)

optimised compute-bound code which uses given resources efficiently should produce a low number of stalls and NOP operations. The static power is reduced by shorter runtimes and translating in a very good energy efficiency. Compute-bound problems are therefore not going to be very efficient on low power systems such as ARMs. Intel's more complex architecture is more preferred.

The LMBench memory benchmark shows the bandwidth of all level caches scales linearly with the frequency of the specific CPU (see Fig. 3 and the same data rearranged in Table 4b). This is expected, as the caches' frequency correspond to the core frequency of both architectures. In the main memory, Haswell looses only 4–5% bandwidth when downscaling the frequency by a factor of 2. The DRAM and memory controller frequency does not scale down with the CPU, which should be a considerable advantage mainly in memory-bound problems. On ARM, however, the DRAM bandwidth starts to decrease significantly once the CPU frequency drops below the DRAM frequency, which is 933 MHz. Above this point, the bandwidth is almost independent on the CPU frequency scaling showing only a slight drop probably due to an imperfect clock divider.

Comparing the maximum overall bandwidth of a dual-socket Haswell, A15 and A7 quadcores, Haswell is 7× faster in L1 cache, 15× faster in the last level cache and 20× faster in the main memory than A15. A15 is 4× faster in L1 cache, 2× faster in L2 cache and 2.5× faster in the main memory than A7. These numbers roughly correspond to the differences in power consumption and theoretical performance of the CPU.

The single-core bandwidth shows that Haswell's performance drops almost 10× across all cache levels and the main memory compared to employing all the 2×6 cores. At least 10 cores is necessary to fully saturate the data transfers. Lowering the frequency also negatively impacts the performance much more prominently. On ARM, the caches' bandwidth drops by a factor of unused cores, however, the main memory bandwidth reaches almost 70% compared to using all cores. Using 2 cores (not shown in the tables) saturates the main memory by almost 90% on both Cortexes. This fact could be exploited on appropriate ARM architectures in heavy memory-bound applications where most of the cores could be switched off to improve energy-efficiency without significant performance penalties.

The overall energy efficiency is presented in Table 4c. As long as data sits in caches, the lowest frequency of Haswell performs the best, followed by A15 on 1400 MHz and A7 on its maximum 1400 MHz. In the main memory, both Cortexes are the most efficient right around the frequency of their DRAM module (933 MHz). Dropping the frequency any lower results in a very poor efficiency mainly because of the increasingly

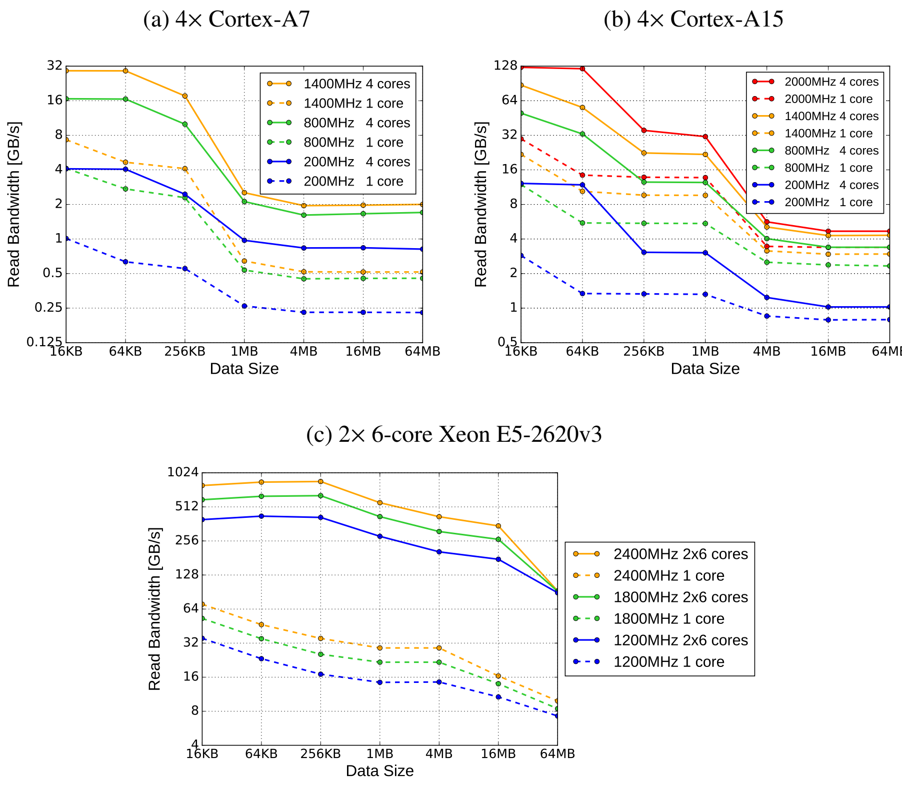

Fig. 3: Bandwidth comparison of memory and cache data read using LMBench.

prominent drop in bandwidth, and also because the static power becoming dominant as the computing time increases.

Table 4d shows the net energy efficiency. This metric almost completely suppresses the effect of static power and computing time on the energy demands. Lower frequencies become more favourable and even the lowest frequency is very often the most efficient one on all architectures. This metric suits both Cortexes better. The more efficient A7 as its overhead of the static power is more prominent due to the whole kit being measured for energy demands whereas only sockets' and DRAMs' hardware counters are taken into account on Haswell.

5.2 Single-Threaded Algorithms

All benchmarks run on a single core only (performance in Fig. 5) and therefore the overhead energy (mainly the static power) becomes much more significant (upper blue-coloured bars in Fig. 6). The overall energy (upper blue and bottom green bar pairs) is the lowest at the maximum frequency on all architectures except A15 being a bit more efficient calculating the Fibonacci and quicksort benchmarks on 1400 MHz. The

(a) Bandwidth (single core)

GB/s	1x Cortex-A7			1x Cortex-A15				1x 1-core Xeon E5-2620v3		
Data Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
16 KB	1.01	4.12	7.32	2.87	11.9	21.8	29.8	35.3	52.9	70.6
64 KB	0.633	2.73	4.66	1.34	5.52	10.4	14.4	23.3	35.0	46.6
256 KB	0.553	2.29	4.10	1.33	5.48	9.63	13.8	17.0	25.5	35.2
1 MB	0.261	0.536	0.643	1.32	5.46	9.60	13.7	14.4	21.7	29.0
4 MB	0.230	0.450	0.518	0.853	2.51	3.15	3.45	14.5	21.7	29.0
16 MB	0.230	0.454	0.516	0.790	2.38	2.95	3.38	10.7	14.0	16.4
64 MB	0.229	0.455	0.516	0.794	2.33	2.96	3.38	7.3	8.4	9.9

(b) Bandwidth (all cores)

GB/s	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Data Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
16 KB	4.08	16.6	29.2	12.2	49.9	87.5	125.0	395	592	790
64 KB	4.05	16.5	29.1	11.8	32.9	55.9	121.7	424	635	847
256 KB	2.45	10.0	17.6	3.06	12.5	22.5	35.3	413	643	858
1 MB	0.974	2.12	2.53	3.04	12.4	21.8	31.2	281	420	557
4 MB	0.835	1.62	1.95	1.24	4.03	5.09	5.65	205	310	419
16 MB	0.838	1.66	1.97	1.02	3.39	4.28	4.68	176	264	348
64 MB	0.816	1.70	2.00	1.02	3.39	4.31	4.68	89.1	91.8	92.5

(c) Overall energy efficiency (all cores)

GB/W	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Data Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
16 KB	2.39	7.92	12.0	3.58	10.8	13.2	9.17	14.9	10.6	8.00
64 KB	2.33	8.30	11.5	3.52	7.12	8.12	9.31	15.5	11.0	8.37
256 KB	1.34	4.77	6.91	0.917	2.94	3.67	3.01	14.9	11.1	8.43
1 MB	0.490	0.913	0.975	0.954	2.73	3.30	2.61	9.14	6.56	4.96
4 MB	0.389	0.701	0.763	0.348	0.865	0.868	0.624	6.04	4.19	3.11
16 MB	0.396	0.777	0.698	0.284	0.694	0.653	0.512	4.10	2.60	2.24
64 MB	0.417	0.745	0.746	0.275	0.740	0.739	0.513	0.411	0.324	0.263

(d) Net energy efficiency (all cores)

GB/W▼	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Data Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
16 KB	31.8	39.9	44.3	22.4	31.2	26.6	12.9	20.9	18.1	16.0
64 KB	39.0	42.7	39.2	23.3	20.8	15.7	13.3	21.7	18.8	16.7
256 KB	12.8	24.2	22.9	6.37	10.2	8.06	4.51	20.9	18.9	16.9
1 MB	2.77	3.33	3.09	9.48	8.21	6.65	3.89	12.8	11.2	9.91
4 MB	1.63	2.60	2.50	1.78	2.48	2.01	1.10	8.45	7.12	6.23
16 MB	1.73	3.66	1.89	1.39	1.83	1.33	0.895	5.74	4.42	4.48
64 MB	2.54	2.82	2.21	1.18	2.20	1.72	0.900	0.644	0.609	0.579

worst 🟥🟧🟨🟩 best (each row coloured separately)

Fig. 4: Memory and cache data read bandwidth and energy efficiency using LMBench.

cause is, similarly to the previous synthetic benchmarks but even more noticeable in this case, the static power of the system. Having only one core working and the other ones idling, the static power becomes the most dominant energy consumer and any decrease in frequency only prolongs the runtime and the system energy overhead problem.

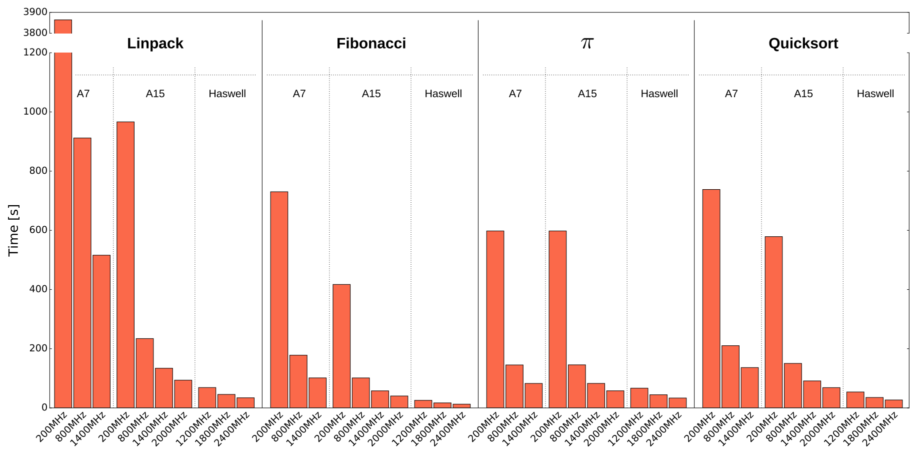

Fig. 5: Performance of single-threaded algorithms - Linpack using 4096 repetitions on 512×512 grid, recursive Fibonacci series calculating the 47th element, π calculation iteratively for 5×10^9 iterations and recursive quicksort sorting 150 000 000 elements.

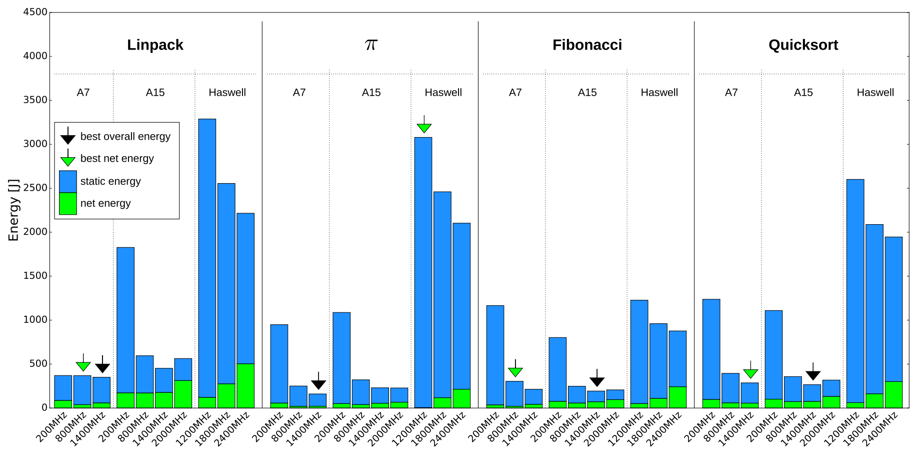

Fig. 6: Energy efficiency of single-threaded algorithms (bottom green part shows the net energy, blue and green together the overall, green arrow points to the lowest net energy of a given benchmark, black arrows points to the lowest overall energy) - Linpack using 4096 repetitions on 512×512 grid, recursive Fibonacci series calculating the 47th element, π calculation iteratively for 5×10^9 iterations and recursive quicksort sorting 150 000 000 elements.

The net energy, shown as green bars, can benefit from frequency scaling because it suppresses the effect of the static power, specifically on Intel, where each frequency

(a) Performance

GFlop/s	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^3 (112 MB)	0.140	0.527	0.847	0.352	1.33	2.10	2.66	29.3	36.8	43.8
256^3 (896 MB)	0.146	0.483	0.755	0.382	1.46	2.08	2.88	33.2	44.3	51.8
257^3 (907 MB)	0.063	0.250	0.446	0.144	0.571	0.955	1.29	2.91	4.87	6.15

(b) Overall energy efficiency

GFlops/W	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^3 (112 MB)	0.074	0.223	0.284	0.148	0.332	0.304	0.174	0.295	0.283	0.290
256^3 (896 MB)	0.074	0.189	0.253	0.163	0.373	0.330	0.200	0.296	0.262	0.229
257^3 (907 MB)	0.035	0.113	0.155	0.065	0.165	0.162	0.096	0.030	0.044	0.042

(c) Net energy efficiency

GFlops/W ▼	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^3 (112 MB)	0.575	0.732	0.681	0.536	0.605	0.429	0.211	0.561	0.445	0.421
256^3 (896 MB)	0.451	0.526	0.608	0.617	0.698	0.485	0.247	0.509	0.363	0.290
257^3 (907 MB)	0.371	0.432	0.390	0.296	0.348	0.246	0.121	0.059	0.077	0.061

worst 🟥🟧🟨🟩 best

Fig. 7: The k-Wave simulation toolbox (the k-space pseudospectral method).

drop results in a linear decrease in power demands. A15 is the most efficient around 800–1400 MHz and A7 at 800 MHz. A7 is overall the most efficient core.

5.3 Multi-Threaded Algorithms

The last group of benchmarks represents parallel algorithms corresponding to common HPC workloads.

The k-Wave toolbox [13] is based on the k-space pseudospectral method, which is characterised by high accuracy, fast convergence and a low number of grid points per wavelength. The 3D fast Fourier transforms are computed using the FFTW[12] 3.3.4 library.

Performance-wise, Haswell achieves more than 50 GFlop/s, which is about 15% of Linpack's performance. A15 is capable of almost 2.9 GFlop/s, almost a third of Linpack, and A7 achieves 0.85 Gflop/s, almost half of Linpack's performance. The CPUs are limited by the main memory bandwidth, and as the less powerful architecture is used, the performance is much closer to its theoretical limit. This can be observed even more prominently on the FDTD method.

Table 7b shows all three CPUs behaving similarly in terms of the peak energy efficiency, achieving around 0.3 GFlops/W, A15 being the most efficient running at 800 MHz. Intel's efficiency drops considerably using the prime domain size of 257^3 (which is the worst case scenario and should be avoided), breaking the vectorisation and memory access patterns. Since the code is unable to be vectorised on ARM, because the FFTW library did not provide a support for Neon vectorisation at the time of writing this paper, A15's and A7's drops are not so significant.

[12] http://www.fftw.org/

Table 8: Performance (double precision), overall and net energy efficiency comparison using the LAMMPS molecular dynamics simulator and its Fene benchmark.

	4x Cortex-A7			4x Cortex-A15				2x 6-core Xeon E5-2620v3		
	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
GFlop/s	0.046	0.142	0.191	0.080	0.265	0.374	0.439	2.93	4.21	5.10
GFlops/W	0.023	0.057	0.062	0.037	0.073	0.067	0.039	0.042	0.044	0.042
GFlops/W▼	0.132	0.177	0.140	0.130	0.136	0.103	0.051	0.125	0.094	0.070

worst — best (each row coloured separately)

k-Wave's net energy efficiency (Table 7c) shows A7 as the most efficient at 800 MHz, followed by A15 also at 800 MHz and Haswell at 1200 MHz. Haswell is much closer to ARM compared to the FDTD method mainly because the raw performance is much higher using k-Wave.

LAMMPS is the only benchmark presented using MPI instead of a threading and double precision floating point arithmetic. The main difference is that ARMv7 does not support the double precision vectorisation, its 128-bit NEON registers support only a single precision. LAMMPS is a typical example of a problem with mutual interactions of a high number of independent and relatively simple elements. The performance of this algorithm class is often lower, however it should theoretically benefit even more from frequency scaling and the usage of low power architectures.

Performance-wise (see Table 8), Haswell achieves more than 5 GFlop/s, while A15 is about 11× slower and A7 is roughly 25× slower. Overall energy efficiency on A15 and A7 is about 1.5–2× better than on Haswell, which is the biggest difference of all the presented benchmarks. Net energy efficiency is the best on A7, followed by A15 and Haswell being the least efficient.

The FDTD's performance overview is shown in Table 9a. FDTD is an example of a memory-bound problem because the number of operations per one byte of data is relatively low characterised by local data sharing only (no global information is needed), good scalability and a low number of cache misses.

In terms of performance, the dual-socket Haswell is almost 4–5× faster than A15, which is 2-3× more powerful than A7. While in the HPL benchmark, Haswell was more than 30× faster than A15, the difference in memory-bound applications shrinks quite dramatically. The 512^2 domain size does not fit into the L2 cache (two separate matrices are allocated for even and odd iterations and two matrices for heat conductivity properties of each point) of both Cortexes (2 MB) and the performance drop is quite radical. However, in the case of Haswell, exceeding the L3 cache size (15 MB) with the 1024^2 domain size does not hinder the performance. The 2048^2 and 4096^2 sizes were also tested (not shown in the table for the sake of brevity) and the performance stayed around the 20 GFlop/s mark, most likely because of the help of prefetcher.

Table 9b displays the energy efficiency across all CPUs and domain sizes. A7 is the most efficient, capable of more than 0.6 GFlops/W running at 1400 MHz. A15's best frequency is between 800–1400 MHz, for Intel it is its highest - 2400 MHz.

The net energy efficiency, presented in Fig. 9c, shows A7 as the most efficient reaching 2.2 GFlops/W running at 800 MHz. A15 is about half as efficient as A7, running the

(a) Performance

GFlop/s	**4x Cortex-A7**			**4x Cortex-A15**				**2x 6-core Xeon E5-2620v3**		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^2 (256KB)	0.238	0.990	1.75	0.475	1.95	3.42	4.81	10.2	12.9	16.0
256^2 (1MB)	0.219	0.846	1.34	0.479	1.98	3.44	4.91	11.5	16.8	22.3
512^2 (4MB)	0.171	0.637	1.04	0.320	1.24	1.89	2.20	10.5	16.1	20.0
1024^2 (16MB)	0.112	0.439	0.731	0.209	0.802	1.29	1.53	9.15	15.0	19.8

(b) Overall energy efficiency

GFlops/W	**4x Cortex-A7**			**4x Cortex-A15**				**2x 6-core Xeon E5-2620v3**		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^2 (256KB)	0.135	0.485	0.651	0.223	0.561	0.551	0.338	0.156	0.164	0.173
256^2 (1MB)	0.120	0.396	0.470	0.222	0.581	0.547	0.319	0.174	0.210	0.234
512^2 (4MB)	0.097	0.301	0.366	0.140	0.343	0.319	0.178	0.162	0.201	0.208
1024^2 (16MB)	0.061	0.205	0.257	0.094	0.237	0.241	0.142	0.140	0.185	0.200

(c) Net energy efficiency

GFlops/W▼	**4x Cortex-A7**			**4x Cortex-A15**				**2x 6-core Xeon E5-2620v3**		
Grid Size	200MHz	800MHz	1400MHz	200MHz	800MHz	1400MHz	2000MHz	1200MHz	1800MHz	2400MHz
128^2 (256KB)	1.07	2.20	1.76	1.07	1.19	0.819	0.413	0.527	0.454	0.375
256^2 (1MB)	0.787	1.55	1.15	1.03	1.26	0.807	0.383	0.576	0.563	0.490
512^2 (4MB)	0.766	1.20	0.902	0.538	0.698	0.485	0.226	0.559	0.539	0.434
1024^2 (16MB)	0.363	0.797	0.633	0.397	0.518	0.389	0.186	0.472	0.483	0.403

worst 🟥🟨🟩 best

Fig. 9: The Finite Difference Time Domain method - 4th order in space, 1st order in time.

most efficiently at 800 MHz. Intel is the least efficient giving best results at its lowest frequency - 1200MHz.

6 Conclusion

In this paper, the effect of voltage and frequency scaling on performance and energy efficiency was studied comparing the Intel Xeon Haswell and ARMv7 big.LITTLE architectures. Two different techniques for measuring energy efficiency were presented, the overall and the net, isolating only the algorithm's energy demands.

The results showed that frequency scaling can bring significant energy savings mainly on the ARM architecture (1.5–2× on the optimal frequency compared to the maximum one). While Intel processors can also benefit from the frequency scaling, the profit is not so significant due to the higher energy demands for the rest of the system (the static power), mainly the DRAM modules and also lower flexibility regarding the frequency range.

The Samsung Odroid-XU4 board on the other hand provides a very flexible range of frequencies and a much lower energy overhead required to power the system around the processor. Overall, the lower range of frequencies does not prove to be efficient on any set of benchmarks. The "sweet spot" for both the Cortex-A7 and Cortex-A15 quadcores lies around the frequency of its DRAM, which is 933 MHz, providing better energy efficiency than the Haswell processors.

In High Performance Linpack, the peak performance of the dual-socket Haswell is 30× better than the Cortex-A15 quadcore. However, in parallel scientific applications, the difference shrank to about 5–15×, which results in a better performance to purchase price ratio in favour of ARM ($70 for the ARM kit vs. 2× $500 for only the Haswell processors).

Table 3 presents a 10-year lifetime comparison of all architectures running the LAMMPS simulator using the most energy-efficient setting. For ~30× more energy consumed, Haswell provides ~20× better performance.

Table 3: 10-year lifetime comparison running LAMMPS using the most overall energy-efficient setting (A7 1400 MHz, A15 800 MHz, Haswell 1800 MHz), considering €0.2 for 1 kWh.

Processor	Energy [MJ]	Electricity costs [€]	PFlops
4× Cortex-A7	0.975	194	60.2
4× Cortex-A15	1.16	232	83.6
2×6 Haswell	29 900	5 970	1 330

Table 4: Comparison of the overall energy efficiency and performance relative to 4× Cortex-A15. For each benchmark and architecture, the most energy efficient frequency is chosen. The same frequency is then used to compare performance (higher number is better).

Algorithm	Overall energy efficiency			Performance		
	4×A7	4×A15	2×6 Haswell	4×A7	4×A15	2×6 Haswell
HPL	0.496	1	1.67	0.415	1	52.6
LMBench 16 KB (L1)	0.909	1	1.13	0.334	1	4.51
LMBench 1 MB (L2)	0.295	1	1.83	0.116	1	9.41
LMBench 64 MB (Main)	1.01	1	0.555	0.588	1	26.3
Linpack (single core)	1.29	1	0.204	0.259	1	3.04
Fibonacci (single core)	1.42	1	0.109	0.398	1	3.18
π (single core)	0.904	1	0.221	1.01	1	2.47
Quicksort (single core)	0.931	1	0.137	0.668	1	3.38
k-Wave 128^3	0.855	1	0.889	0.637	1	22.5
k-Wave 256^3	0.678	1	0.796	0.517	1	22.7
k-Wave 257^3	0.939	1	0.267	0.781	1	8.53
LAMMPS	0.849	1	0.603	0.721	1	15.9
FDTD 128^2	1.16	1	0.308	0.897	1	8.21
FDTD 256^2	0.809	1	0.403	0.677	1	11.3
FDTD 512^2	1.07	1	0.606	0.839	1	16.1
FDTD 1024^2	1.07	1	0.833	0.567	1	15.3
Average	0.918	1	0.660	0.589	1	14.1

Table 4 shows a complete comparison of all architectures and benchmarks relative to 4× Cortex-A15. For each benchmark, each architecture runs at the most overall energy-efficient frequency, and then, using the same frequency, the performance is compared. For example, in the HPL benchmark Haswell provides 1.67× more GFlops per Watt, while the performance is 52.6× better (251 GFlop/s at 1800 MHz vs. 4.77 GFlop/s at 800 MHz). The dual-socket Haswell is the most energy efficient in the synthetic benchmarks - HPL and LMBench, on average the efficiency is 0.66× worse than A15's for 14.1× better performance. Overall, Odroid is more energy-efficient in most of the presented benchmarks, but for the price of a significant performance drop.

The results presented in this paper can be used to save energy on similar systems. However, our study focuses only on single shared memory "nodes", leaving further measurements of power and energy on similar distributed systems for future work, which will focus on distributed ARMv8 clusters provided by the Mont-Blanc project.

7 Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science LQ1602. This work was also supported by the FIT-S-17-3994 Advanced parallel and embedded computer systems project.

References

1. Dynamic voltage and frequency scaling based on workload decomposition. In: Low Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on. pp. 174–179 (Aug 2004)
2. Cha, K.: Performance evaluation of lammps on multi-core systems. In: High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC). pp. 812–819 (Nov 2013)
3. Choi, J., Dukhan, M., Liu, X., Vuduc, R.: Algorithmic time, energy, and power on candidate hpc compute building blocks. In: Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium. pp. 447–457. IPDPS '14, IEEE Computer Society, Washington, DC, USA (2014), http://dx.doi.org/10.1109/IPDPS.2014.54
4. Davies, T., Karlsson, C., Liu, H., Ding, C., Chen, Z.: High performance linpack benchmark: A fault tolerant implementation without checkpointing. In: Proceedings of the International Conference on Supercomputing. pp. 162–171. ICS '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1995896.1995923
5. Hackenberg, D., Schne, R., Ilsche, T., Molka, D., Schuchart, J., Geyer, R.: An energy efficiency feature survey of the intel haswell processor. In: Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International. pp. 896–904 (May 2015)
6. hsing Hsu, C., chun Feng, W.: A power-aware run-time system for high-performance computing. In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference. pp. 1–1 (Nov 2005)
7. Huang, S., Lang, M., Pakin, S., Fu, S.: Measurement and characterization of haswell power and energy consumption. In: Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing. pp. 7:1–7:10. E2SC '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2834800.2834807

8. Liang, W.Y., Chen, S.C., Chang, Y.L., Fang, J.P.: Memory-aware dynamic voltage and frequency prediction for portable devices. In: 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. pp. 229–236 (Aug 2008)
9. McVoy, L., Staelin, C.: Lmbench: Portable tools for performance analysis. In: Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference. pp. 23–23. ATEC '96, USENIX Association, Berkeley, CA, USA (1996), http://dl.acm.org/citation.cfm?id=1268299.1268322
10. Nikola Rajovic, e.a.: The mont-blanc prototype: An alternative approach for hpc systems. SC16 (2016)
11. Schuchart, J., Gerndt, M., Kjeldsberg, P.G., Lysaght, M., Horák, D., Říha, L., Gocht, A., Sourouri, M., Kumaraswamy, M., Chowdhury, A., Jahre, M., Diethelm, K., Bouizi, O., Mian, U.S., Kružík, J., Sojka, R., Beseda, M., Kannan, V., Bendifallah, Z., Hackenberg, D., Nagel, W.E.: The readex formalism for automatic tuning for energy efficiency. Computing pp. 1–19 (2017), http://dx.doi.org/10.1007/s00607-016-0532-7
12. Spiliopoulos, V., Kaxiras, S., Keramidas, G.: Green governors: A framework for continuously adaptive dvfs. In: Green Computing Conference and Workshops (IGCC), 2011 International. pp. 1–8 (July 2011)
13. Treeby, B.E., Cox, B.T.: k-wave: Matlab toolbox for the simulation and reconstruction of photoacoustic wave-fields. J. Biomed. Opt. 15(2), 021314 (2010)
14. Weaver, V.M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., Moore, S.: Measuring energy and power with papi. In: 2012 41st International Conference on Parallel Processing Workshops. pp. 262–268 (Sept 2012)