Performance and Energy Metrics for Multi-threaded Applications on DVFS Processorss

Thomas Rauber
Computer Science Department,
Universität Bayreuth,
95440 Bayreuth, Germany
rauber[at]uni-bayreuth.de

Gudula Rünger
Computer Science Department,
Technische Universität Chemnitz,
09107 Chemnitz, Germany
ruenger[at]cs.tu-chemnitz.de

Matthias Stachowski
Computer Science Department,
Universität Bayreuth,
95440 Bayreuth, Germany
stachowski[at]uni-bayreuth.de

To cite this version:
Thomas Rauber, Gudula Rünger, Matthias Stachowski: Performance and Energy Metrics for Multi-threaded Applications on DVFS Processors. Sustainable Computing: Informatics and Systems, Vol. 17: pp. 55-68. Elsevier – ISSN 2210-5379, 2017 (released März 2018). DOI: 10.1016/j.suscom.2017.10.015

Abstract

Due to their internal execution characteristics, application programs exploit the hardware very differently, which leads to a quite diverse behavior concerning their performance or the energy consumed for their execution. A change of the operational frequency of DVFS processors leads to further variations in performance and energy consumption, as does the exploitation of thread parallelism on multicores. This article combines frequency scaling and thread-parallelism and considers several new metrics for the evaluation of an application's performance and energy consumption. As application programs, the PARSEC benchmark suite and the SPLASH-2 benchmark suite are investigated. The PARSEC benchmark suite provides an up-to-date collection of applications with different workloads on chip-multiprocessors. The SPLASH-2 is a common suite for scientific studies on parallel shared memory machines. Intel Core i7 processors are used as hardware platforms for the evaluation.

1 Introduction

A low energy consumption as well as a good performance are important properties of today‘s application codes. These properties are influenced by the software structure, the execution characteristics, and the way in which the code exploits the hardware details of the execution platform. The internal organization of the processor architecture may have a large influence on the resulting runtime and energy consumption, depending on how well the operations defined by the application code can be mapped onto the functional units of the processor. Also, recent processors provide features, such as chip-multiprocessing or dynamic voltage frequency scaling (DVFS), which can be exploited to achieve a better performance or energy balance. However, it is not a priori predictable whether the use of more threads leads to a smaller execution time of an application and whether the use of a smaller operational frequency leads to a reduction in the energy consumption. Instead, for a given application there is typically an optimal number of threads beyond which the execution time cannot be reduced further. Similarly, there is often an optimal operational frequency beyond which a further frequency reduction does not lead to a further energy reduction.

In this article, we study the combination and interaction of energy efficiency and runtime performance for multi-threaded applications on multicore DVFS processors. We consider the dependence of energy consumption as well as runtime performance from both, the number of threads used and the operational frequency chosen. The runtime performance usually improves with an increasing number of threads but decreases with a down-scaling of the frequency. Also the energy used for an application varies when varying the number of threads or when scaling the frequencies. And although the energy consumed depends on the execution time, it can be observed that the lowest execution time achieved for a specific setting of number of threads and operational frequency does not necessarily lead to the lowest energy consumption. This is caused by an application-specific power consumption.

The diverse interactions of the effects of a varying number of threads and varying frequencies on the runtime performance and the energy consumption of applications can be quite intricate. Thus, the question arises whether appropriate metrics can be defined which can capture the dependence of performance and energy consumption on the two varying parameters, i.e., the number of threads and the operational frequency, and how an improvement effect can be expressed. Since both runtime performance and energy consumption are consided together, a metrics combining them is needed. In this article, we propose several metrics, including speedup and reduction factors for the execution time and the energy in isolation as well as combined metrics. All these metrics depend on two variables, which are the number of threads and the frequency. Also, metrics for the power consumption are introduced with which application-specific power values can be assessed.

The Princeton Application Repository for Shared-Memory Computers (PARSEC) [6] is a collection of multi-threaded benchmarks with different parallel workloads and execution characteristics. PARSEC provides programs from a wide range of applications based on the latest techniques in the specific domains and with an emphasis on large workloads as well as multicore parallelism. The contribution concerning the PARSEC benchmarks has two aspects. First, we demonstrate and evaluate our metrics with respect to their expressiveness and validity. Second, we provide an intensive study of the performance and energy characteristics of the PARSEC benchmarks on recent Intel multicore processors that adds another study to the investigation of the PARSEC benchmarks. The results concerning energy might be valuable for users of the benchmark suite.

The SPLASH-2 benchmark suite [32] is a popular benchmark suite comprising eleven applications mainly from scientific computing and graphics, which were the main application areas for parallel programming when the SPLASH-2 benchmark suite has been assembled. The benchmarks have been chosen and optimized for parallel shared memory machines of the 90th. Thus, the optimizations naturally address different hardware features then are now availbale in recent multicore architectures. Nevertheless, the programming model of the SPASH-2 benchmarks is suitable for multicore programming and it is worthwhile to study these application codes with respect to multicore parallelism and frequency scaling, which will gives insight into the behavior of scientific applications, which are usually used for a longer period, on recent architectures.

The contribution of this article comprises an intensive empirical investigation of the performance and the energy behavior of multithreaded applications with respect to varying numbers of cores and varying frequencies on recent multicore DVFS processors. As hardware platform, we use two Intel Core i7 processors from the 4th (Haswell) and the 6th (Skylake) generation, which both provide frequency scaling, and an ARM big.LITTLE architecture with an Samsung Exynos5422 Cortex A15 2.0 GHz quadcore as well as a Cortex 1.4 GHz A7 quadcore CPU, provided by an Odroid XU4 Board. The study gives insight into the performance and energy characteristics of the PARSEC and the SPLASH-2 benchmarks, thus providing results for application optimized for recent as well as for older parallel shared memory machines. The investigations and evaluations have been done with the help of well-known and also newer metrics capturing both criteria, energy and performance.

The rest of the article is structured as follows: Section 2 describes the energy model and the evaluation strategy used. Section 3 presents the new metrics. Section 4 evaluates of the PARSEC and SPLASH-2 benchmarks with these metrics and discusses the results. Section 5 discusses related work and Section 6 concludes the article.

2 Multivariable energy model

The execution time $T[sec]$ of an application code varies with the number of threads p used for the execution and the operational frequency f chosen, as does the power drawing $P[Watt]$. To express the dependency of the energy consumption $E = T \cdot P[Joule]$ on the number of threads and the operational frequency, we develop an energy model which explicitly uses the two variables p and f.

2.1 Energy model for frequency scaling

When considering the energy consumption of application programs for DVFS processors, it is useful to introduce frequency scaling factors as follows: The frequency scaling for a DVFS processor is expressed by a dimensionless scaling factor $s \geq 1$ which describes a smaller frequency $\tilde{f} < f_{max}$ as $\tilde{f} = f_{max}/s$ where f_{max} is the maximum frequency possible for the processor.

Power models for DVFS processors distinguish the dynamic power consumption and the static power consumption. The dynamic power consumption $P_{dyn}(f)$ is related to the supply voltage and the switching activity during the computing activity of the processor and can be expressed by $P_{dyn}(f) = \alpha \cdot C_L \cdot V^2 \cdot f$ where α is the switching probability, C_L is the load capacitance, and V is the supply voltage. The frequency f depends linearly on the supply voltage V, i.e., $V = \beta \cdot f$ with some appropriate constant β. Thus, the dependence of the dynamic power consumption on the frequency f can be expressed as $P_{dyn}(f) = \gamma \cdot f^3$ with $\gamma = \alpha \cdot C_L \cdot \beta^2$ or when using the corresponding scaling factor s as $P_{dyn}(s) = s^{-3} \cdot P_{dyn}(1)$ where $P_{dyn}(1)$ is the dynamic power consumption in the un-scaled case.

The static power consumption $P_{stat}(f)$ is intended to capture the leakage power consumption as well as the power consumption of peripheral devices and can be expressed by $P_{stat}(f) = V \cdot N \cdot k_{\text{design}} \cdot I_{\text{leak}}$, where N is the number of transistors, k_{design} is a design dependent parameter, and I_{leak} is a technology-dependent parameter [7]. Again using $V = \beta \cdot f$ leads to a linear dependence of the static power on f, i.e., $P_{stat}(f) = \delta \cdot f$ with $\delta = N \cdot k_{\text{design}} \cdot I_{\text{leak}} \cdot \beta$ or $P_{stat}(s) = s \cdot P_{stat}(1)$ where $P_{stat}(1)$ is the static power consumption in the un-scaled case. Other authors have also proposed to make the simplified assumption that P_{static} is independent of the voltage or frequency scaling [33], i.e., $P_{stat}(s) = P_{stat}(1) = P_{static}$. In the following, we use this assumption. The total power consumption includes both power components $P_{dyn}(s)$ and P_{static}.

Reducing the operational frequency of a processor by a scaling factor of s usually decreases the power consumption, however, it increases the execution time $T(1)$ of an application program by the same factor compared to an un-scaled execution, i.e., $T(s) = s \cdot T(1)$. Using the power and the execution time depending on the scaling used for the execution of the application program leads to an energy model depending on the scaling factor s:

$$\begin{aligned} E(s) &= (P_{dyn}(s) + P_{static}) \cdot s \cdot T(1) \\ &= (s^{-3} \cdot P_{dyn}(1) + P_{static}) \cdot s \cdot T(1) \\ &= (s^{-2} \cdot P_{dyn}(1) + s \cdot P_{static}) \cdot T(1) \end{aligned} \tag{1}$$

2.2 Energy model for parallel execution

The execution of multithreaded programs introduces a further variable into the model, which is the number of threads used for a specific code execution. The execution time usually decreases with an increasing number of threads (typically in a non-linear way) until a saturation point is reached, which strongly depends on the application.

On the other hand, also the power drawing varies with the number of threads, usually in a way that the power drawing increases with an increasing number of threads. Table 1 shows the power consumption and Figure 4 illustrates the energy consumption impacts of multithreading and frequency scaling for the example applications swaption and blackscholtes from the PARSEC suite. The table shows that the power drawing increases with the frequency for a fixed number of threads. For a fixed frequency, the power drawing increases up to four threads, corresponding to the number of physical cores, and then it may increase further or may slightly decrease. The table shows the enormous differences in the power consumption with values between about 5 Watt and 50 Watt. Taking the dependence of the execution time and power drawing on both, the number of threads and the frequency scaling, into consideration results in an energy model for two variables. More precisely, Equ. (1) is extended to take varying numbers p of executing threads into consideration, in addition to the scaling factor s, yielding

$$E(p, s) = (s^{-2} \cdot P_{dyn}(p, 1) + s \cdot P_{stat}(p, 1)) \cdot T_{par}(p, 1) \tag{2}$$

Table 1: Power Consumption in Watt of the PARSEC benchmark swaption on Haswell for different frequencies and different number of threads

	power consumption threads			
Freq	1	2	4	8
0.8 GHz	4.97	6.45	8.75	9.34
1.0 GHz	5.40	7.37	10.17	10.66
1.2 GHz	5.84	8.05	11.32	12.61
1.4 GHz	6.74	9.49	14.37	15.35
1.5 GHz	7.33	10.09	15.18	16.44
1.7 GHz	7.80	11.33	17.20	18.60
1.9 GHz	8.43	12.68	19.00	21.16
2.1 GHz	9.24	13.52	21.82	23.81
2.3 GHz	10.05	14.82	24.18	27.01
2.5 GHz	11.39	17.29	25.27	32.27
2.7 GHz	12.22	19.15	32.31	35.95
2.8 GHz	12.77	20.05	33.90	37.96
3.0 GHz	14.06	22.46	37.52	42.43
3.2 GHz	15.92	25.17	42.71	47.40
3.4 GHz	17.19	27.39	45.87	52.61

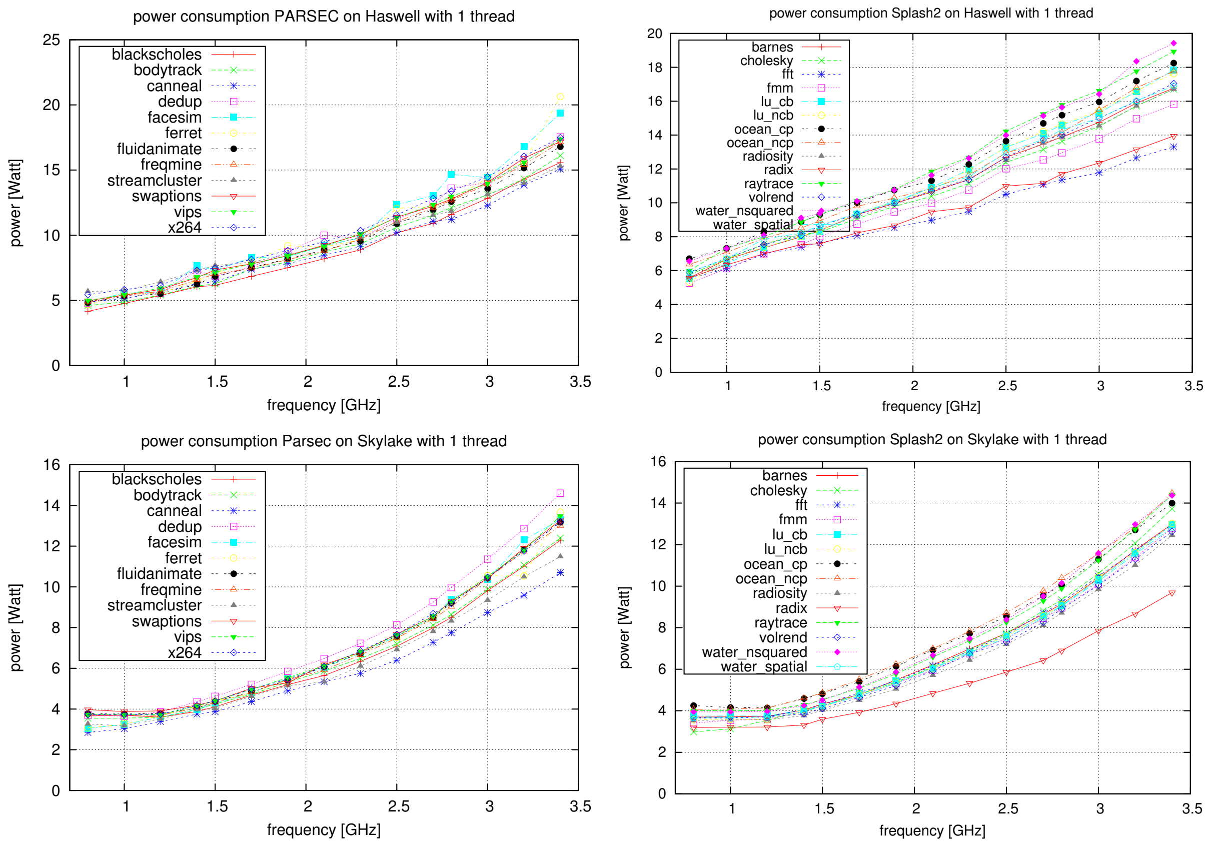


Figure 1: Sequential execution: Power consumption (in Watt) of PARSEC (left) and SPLASH2 (right) benchmarks executed with one thread on an Intel Core i7 Haswell (top) and Core i7 Skylake (bottom) architecture using different frequencies.

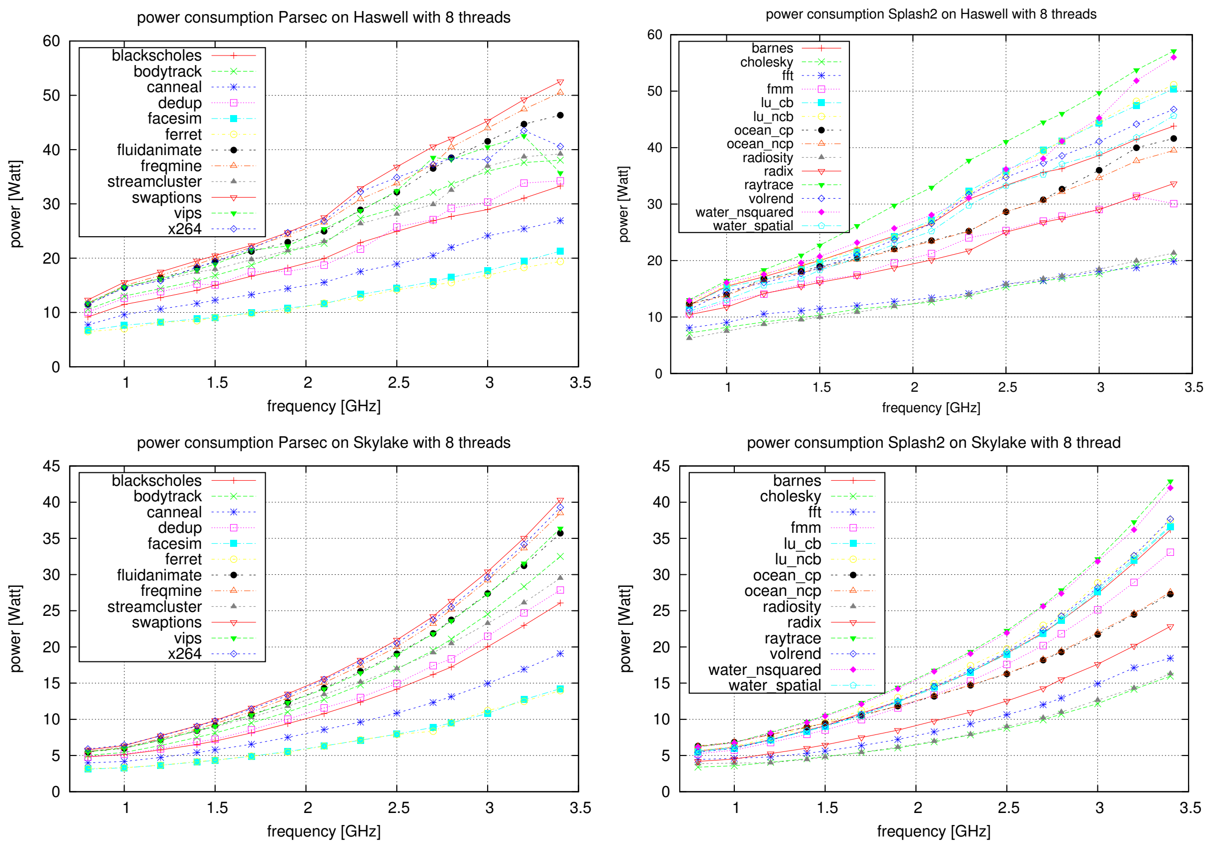


Figure 2: Multi-threaded execution: Power consumption (in Watt) of PARSEC (left) and SPLASH2 (right) benchmarks executed with eight thread on an Intel Core i7 Haswell (top) and Core i7 Skylake (bottom) architecture using different frequencies.

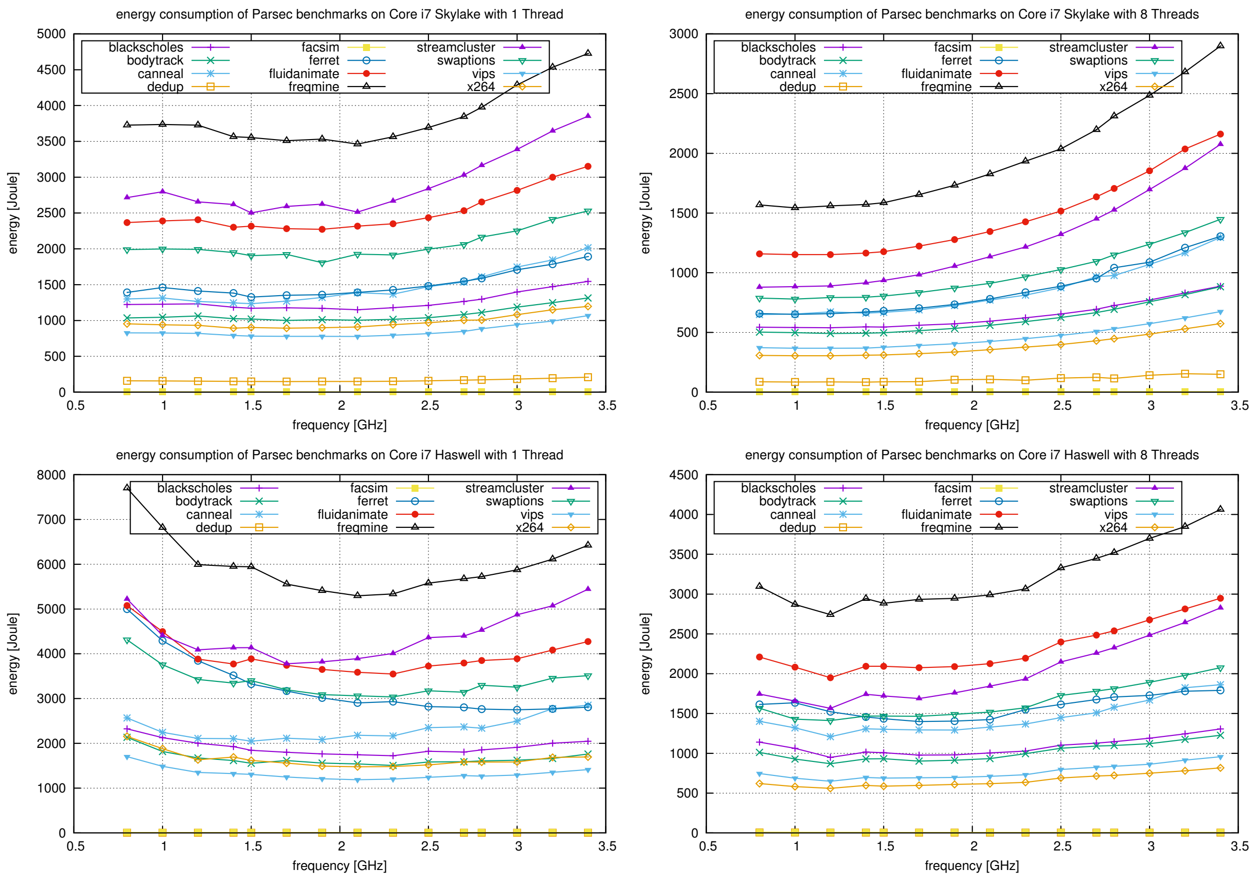


Figure 3: Energy consumption of the PARSEC benchmarks executed with one thread (top) and eight threads (bottom) on an Intel Core i7 Skylake (left) and Haswell (right) architecture using different frequencies.

where $T_{par}(p, 1)$ denotes the execution time with p threads for scaling factor $s = 1$ and $P_{stat}(p, 1)$ denotes the static power consumption when using p threads.

The energy model (2) is a continuously differentiable function in s, which makes it possible to analytically derive an optimal value for the scaling factor s, when p is set to a fixed value. The optimal scaling factor $s_{opt}(p)$ which minimizes $E(p, s)$ can be determined by building the derivative of $E(p, s)$, see also [23]. The resulting optimum scaling factor is

$$s_{opt}(p) = \left(\frac{2 \cdot P_{dyn}(p, 1)}{P_{static}(p, 1)} \right)^{1/3}, \tag{3}$$

assuming that this scaling factor is kept fixed during the execution of the application program. According to Equ. (3), the value of $s_{opt}(p)$ is independent of the actual execution time of the application considered and at first glance seems to be fixed. However, different applications may have different values $P_{dyn}(p, 1)$ and $P_{static}(p, 1)$ for a fixed p due to a different usage of the hardware resources of the processor used. Moreover, using a different number of threads may lead to different values for $P_{dyn}(p, 1)$ and $P_{static}(p, 1)$ due to the exploitation of parallelism. Thus, different applications may have different values for $s_{opt}(p)$ for the same p, and different values for $s_{opt}(p)$ may result for different values of p, even for the same application.

2.3 The PARSEC and SPLASH-2 Benchmark Suites

The PARSEC benchmark suite is a collection of benchmarks aiming at the investigation of thread parallelism on recent chip multiprocessors. In contrast to the earlier very popular SPLASH-2 benchmark suite, the PARSEC benchmarks emphasize on larger workloads typical for today's application programs. Also, while the SPLASH-2 benchmarks primarily come from scientific computing, the PARSEC benchmarks include a broader range of applications, including financial analysis, animation, data mining, or enterprise storage.

The computational characteristics of the PARSEC benchmarks are intensively studied in [6] and [3] where properties concerning the instructions and concerning shared data are investigated. The first properties are the number of floating point operations, ALU instructions, branches and memory accesses. The second kind of properties are the data cache miss rate, the percentage of shared cache lines for reading and for writing, the ratio of memory references reading from shared cache lines and the ratio writing to them. The energy behavior measured with hardware as well as software techniques has been investigated in [24] with an emphasis on frequency scaling. In this article, we extend the investigations of the energy and the execution time depending on the two independent variables frequency and number of threads and especially study newly defined efficiency characteristics, i.e., energy speedup and runtime speedup, as defined in the next section.

The SPLASH-2 benchmark suite is a mature but still very popular benchmark suite released in 1995 and mainly comprising HPC and graphics applications, including Cholesky and LU factorization or raytracing and radiosity algorithms. Due to architectures at that time, the codes have been optimized for multi-nodes with high latencies between nodes, so that communication between them had been avoided when possible.

2.4 Experimental setting

The following experiments have been performed on an Intel Core i7-4770 with the Haswell architecture, providing a maximum frequency of 3.4 GHz and an Intel Core i7-6700 with the Skylake architecture, which also gains from 3.4 GHz of maximum frequency. The usage of Intel Turbo Boost Technology, which would be 3.9 GHz on Haswell and 4.0 GHz on Skylake, was automatically disabled on both systems during the measurements. The reason is that the system call cpufreq-set is needed to fix the operational frequency to a specific value, and this call disables the Turbo Boost feature.

The processors have four physical cores with hyper-threading, leading to eight logical cores. The memory hierarchy includes an 8 MB shared L3 cache as well as a 256 KB L2 cache and a 32 KB L1 cache per core. The main memory size is 16 GB. The specified thermal design power (TDP) is 84 W on the Haswell system and 65 W on the Skylake system.

The time and energy measurements have been performed using the Running Average Power Limit (RAPL) interface and sensors of the Intel architecture [27, 17]. RAPL sensors can be accessed by control registers, known as Model Specific Registers (MSRs), which are updated in intervals of about 1 ms [17]. In particular, we have used the likwid tool-set, especially the likwid-powermeter in Version 4.0 [31], which provides access to the MSRs introduced above. Experiments have shown that the energy measurement with RAPL sensors are quite accurate when compared to measurements with power-meters [27, 25].

The experiments have been performed with the `native` input set of the PARSEC benchmarks, which is the largest input set available and which is intended for performance measurements on real machines [6].

2.5 Power and energy consumption

The power and energy consumption of application programs both depends on the number of threads used and the operational frequency. Figures 1 shows the power consumption in Watt for all PARSEC and SPLASH-2 benchmarks on the Haswell and Skylake processors for an execution with one thread. Figure 2 shows the analogous values for an execution with eight threads. In all cases, the power consumption increases with the operational frequency, as expected, and there is also an increase of the power consumption with the number of threads used. For the highest frequency of 3.4 GHz, the power values are roughly between 10 and 14 Watt on the Skylake system (bottom diagrams in Fig. 1) and between 14 and 20 Watt on the Haswell system (top diagrams in Fig. 1) for both the PARSEC and SPLASH-2 benchmarks. The SPLASH-2 benchmarks show a slightly larger variance of the power consumption than the PARSEC benchmarks especially on the Haswell system. For an execution on eight threads, the power values are between 15 and 40 Watt on the Skylake system and between 20 and 55 Watt on the Haswell system for the highest frequency.

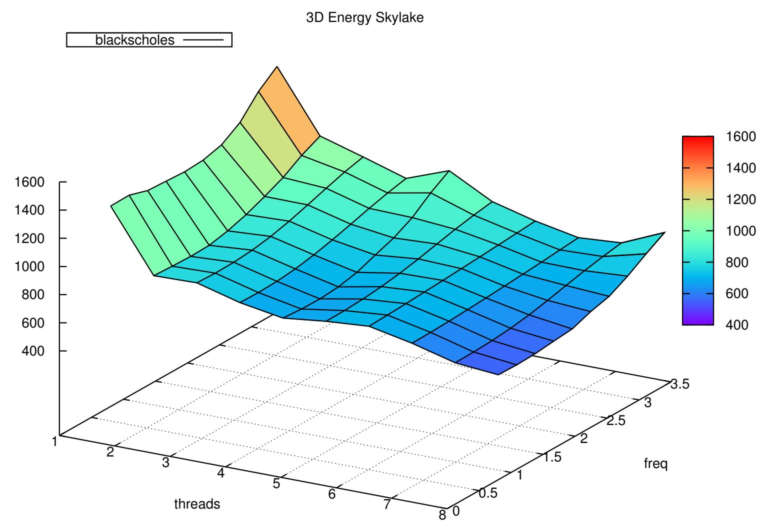


Figure 4: Energy surface plot for the blackscholes benchmark with different frequencies and different number of threads on an i7 Skylake.

Comparing the Haswell and Skylake systems, it can be observed that the power increase with the operational frequency is more or less linear for the Haswell, whereas the Skylake exhibits a more than linear power increase with the frequency, i.e., decreasing the frequency results in larger power savings on the Skylake than on the Haswell system. The qualitative behavior is quite similar for the PARSEC and SPLASH-2 benchmarks, but specific applications may change their power consumption behavior in different ways: E.g. the benchmark `facesim` has a slightly lower power consumption on eight threads than on one thread with the effect that `facesim` has one of the highest power consumptions on one threads compared to the other applications but has the lowest power consumption on eight threads compared to the other applications. Another example is the benchmark `ferret`, which has a higher power comsumption on the Haswell architecture than almost every other benchmark when using only one thread. This difference disappears more and more when the number of threads increases.

Figure 3 shows the energy consumption corresponding to the power consumption of the PARSEC application programs from Figs. 1 and 2. The diagrams show that for most of the programs, there is an operational frequency for which the energy consumption is at a minimum. This effect is especially recognizable for the Haswell architecture, see Fig. 3 (right), but is also visible for Skylake system. The U-shape of the energy curves confirms the energy model described in Subsection 2.1, which predicts such a frequency with minimum energy consumption in Formula (3).

The energy model presented in Subsection 2.1 mainly emphasizes on the minimization of the energy by choosing the appropriate frequency f_{max}/s_{opt}. In Subsection 2.2 we have extended the energy model by including the number of threads as another independent variable in the terms of the formula. However, so far, the number of threads is set to a fixed value so that a family of curves, each curve for a fixed number of threads having its own minimum, is considered. In the next section, we will introduce metrics which also take the number of threads into account to interrelate these curves. Also, we will include the performance and speedup into the consideration.

3 Energy and Performance Metrics

To capture the effect on the performance or energy consumption when using more threads or reducing the frequency, several metrics have been proposed in the literature. Metrics for evaluating the runtime performance depending on the number of cores include the traditional speedup, the number of floating point operations per second (flop/s) or just the total execution time needed to solve a specific problem (time-to-solution). Metrics for evaluating the power or energy efficiency include the performance per Watt and the amount of energy needed to solve a problem (energy-to-solution). An attempt to combine the runtime performance and energy consumption in a single metrics is the energy-delay product (EDP).

In the following, we consider DVFS systems with p_{max} cores and frequencies f between the minimum frequency f_{min} and maximum frequency f_{max}. For parallel programs, the parallel execution time is still an important factor, so that the simultaneous investigation and analysis of energy and performance is relevant. Due to the fact that both evaluation criteria for application codes depend on the two independent variables (number of threads, frequency scaling), the evaluation function is fully described by a two-dimensional function

$$(E,T) : [1, p_{max}] \times [1 : f_{max}/f_{min}] \subset \mathbb{R}^2 \to \mathbb{R}^2$$

with

$$(p, s) \mapsto (E(p, s), T(p, s))$$

and $E(p, s)$ according to Equ. (2) or with measured values.

For each application, these performance functions can be visualized by two surfaces over a grid of (p, s)-points. Such an energy surface for the PARSEC application blackscholes is shown in Fig. 4. When investigating the performance properties of an application, the structure of both surfaces is analyzed for either getting a minimum of E or of T or of a combination of both. For a simpler description of the performance surfaces, we propose and study several metrics, which concentrate on specific performance aspects.

3.1 Performance metrics

The consideration in the last section has assumed that an application program has been executed with a fixed number of threads. In the following, we consider the number of threads as an additional variable. For a specific application program, the sequential execution time for a specific frequency f with scaling factor s is denoted by $T_{seq}(s)$ and the parallel execution time using p threads is denoted by $T_{par}(p, s)$. The *(runtime) speedup* obtained with p threads for a fixed frequency f with scaling factor s is traditionally defined as

$$S(p, s) = \frac{T_{seq}(s)}{T_{par}(p, s)}, \qquad \text{with } s \text{ fixed}, \tag{4}$$

expressing the relative saving of execution time that can be obtained by using p threads instead of one thread for a fixed frequency [22]. For different operational frequencies, different speedups may result, as can be seen in the following sections.

For DVFS processors, the execution time also varies for varying frequencies and increases when the frequency is reduced. To capture how intesively the execution times varies, we define a metrics with a similar flavor as the speedup. This is the *runtime reduction factor*:

$$R(p, s) = \frac{T_{par}(p, s)}{T_{par}(p, 1)}, \qquad \text{with } p \text{ fixed}. \tag{5}$$

The runtime reduction factor describes the relative increase of the execution time that arises when the operational frequency $f = f_{max}/s$ is used instead of f_{max} for a fixed number p of threads. Equ. (5) can also be used for the sequential case, using $T_{seq}(s) = T_{par}(1, s)$. Table 2 shows the reduction factors for the blackscholes benchmark on the Skylake architecture as a typical example.

3.2 Energy metrics

The energy consumption of the execution of an application code also changes with varying frequencies and/or numbers of threads. This is not only an effect of the changes in the execution time but also due to the changes in the power consumption of the individual application code. Thus, it is reasonable to quantify the changes in the

Table 2: Runtime reduction factor for blackscholes on Skylake

	runtime reduction factor			
Freq	1	2	4	8
0.8 GHz	2.748	2.134	3.728	3.738
1.0 GHz	2.782	2.128	2.740	3.362
1.2 GHz	2.774	1.777	2.804	2.821
1.4 GHz	2.426	1.524	2.008	2.403
1.5 GHz	2.259	1.421	1.905	2.245
1.7 GHz	1.998	1.260	1.623	1.985
1.9 GHz	1.788	1.126	1.457	1.776
2.1 GHz	1.612	1.017	1.317	1.612
2.3 GHz	1.474	0.929	1.466	1.474
2.5 GHz	1.357	0.856	1.107	1.352
2.7 GHz	1.256	0.792	1.249	1.258
2.8 GHz	1.212	0.764	1.207	1.219
3.0 GHz	1.132	0.713	1.131	1.130
3.2 GHz	1.063	1.068	0.864	1.059
3.4 GHz	1.000	1.000	1.000	1.000

energy consumption $E(p, s)$ using p threads and frequency f with scaling factor s by a metrics. Analogously to the runtime speedup $S(p, s)$, we define the *energy speedup*

$$ES(p, s) = \frac{E(1, s)}{E(p, s)}, \qquad \text{with } s \text{ fixed.} \tag{6}$$

The energy speedup expresses the relative difference of the energy consumption that occurs by using p threads compared to one thread for a fixed frequency. Figure 3 shows that the energy consumption decreases when using a larger number of threads, i.e., it is expected that $ES(p, s) > 1$.

When considering a varying frequency and its effect on the energy consumption, we propose a metrics similar to the runtime reduction factor $R(p, s)$. We define the *energy reduction factor* as

$$ER(p, s) = \frac{E(p, s)}{E(p, 1)}, \qquad \text{with } p \text{ fixed.} \tag{7}$$

The energy reduction factor expresses the relative difference of the energy consumption when using frequency $f = f_{max}/s$ instead of f_{max} for a fixed number of threads.

3.3 Energy-Delay product

The energy-delay product EDP is defined as the energy consumed by an application program multiplied by its execution time [26]. When considering the scaling factor s explicitly, the EDP is expressed as

$$\begin{aligned} EDP(p, s) &= E(p, s) \cdot (T(p, 1) \cdot s) \qquad [Watt \cdot sec^2] \\ &= (s^{-1} \cdot P_{dyn}(p, 1) + s^2 \cdot P_{static}(p)) \cdot T(p, 1)^2 \end{aligned} \tag{8}$$

The EDP is a single metric that combines effects of execution time and energy consumption for different configurations of an application and captures the translation of energy into useful work. The analytical minimization of the function $EDP(s)$ yields the optimum scaling factor

$$s_{opt}^{EDP}(p) = \left(\frac{P_{dyn}(p, 1)}{2 \cdot P_{static}(p, 1)} \right)^{1/3} \tag{9}$$

for the EDP. It is $s_{opt}^{EDP}(p) < s_{opt}(p)$ from Equ. (3), i.e., to minimize the EDP, a smaller scaling factor must be used than for the minimization of the energy consumption.

Table 3: EDP for the blackscholes benchmark on Skylake.

	energy-delay product			
Freq	1	2	4	8
0.8 GHz	408471.95	174496.01	95548.65	61174.88
1.0 GHz	407648.30	175014.38	91240.54	57132.67
1.2 GHz	420990.95	147781.74	75506.57	50443.76
1.4 GHz	361843.89	127790.57	66892.77	45695.50
1.5 GHz	333869.42	119608.92	62906.10	42396.80
1.7 GHz	295863.46	102313.89	56870.55	38661.13
1.9 GHz	263209.61	96868.74	51741.68	34778.67
2.1 GHz	234092.78	89761.25	49321.55	32750.56
2.3 GHz	219116.60	85904.01	46751.87	31338.13
2.5 GHz	207158.99	81970.41	45030.67	30278.64
2.7 GHz	200559.41	81253.48	44338.35	29693.18
2.8 GHz	197998.20	79843.06	44165.68	30576.95
3.0 GHz	198941.88	80629.03	44707.31	29724.46
3.2 GHz	196821.34	80533.64	44755.90	30045.83
3.4 GHz	194507.31	80398.78	44891.82	30165.26

The significance of the energy delay product can be seen when looking at the energy efficiency (EE). The energy efficiency of an application is defined as performance (flop/s) per energy unit ($Watt \cdot s$) and is measured as $flop/(Watt{\cdot}sec^2)$. Given two EDP values EDP_1 and EDP_2 with $EDP_1 < EDP_2$ yields $1/EDP_1 > 1/EDP_2$, both sides measured in $1/(Watt \cdot sec^2)$, i.e., smaller EDP values indicate a better energy efficiency, and thus, a larger performance per energy unit. Therefore, the EDP is a good measure for the energy efficiency of an application, and s_{opt}^{EDP} optimizes the energy efficiency of an application program for a given execution platform. Table 3 gives the EDP for the blackscholes benchmark on Skylake as example. Similarly to the energy and the parallel execution time, the EDP is a function depending on two variables p and f, merging the two former functions. In order to further evaluate the behavior of this hybrid function, we propose a set of new metrics involving energy, power and executions time in the following subsections.

3.4 Combining energy consumption and speedup

A metric to compare the energy consumption and the (runtime) speedup of a specific application program is *energy per speedup*

$$EPS(p,s) = \frac{E(p,s)}{S(p,s)} \qquad [J] \tag{10}$$

with s fixed, since the speedup is defined for a fixed s, see Equ. (4). The energy per speedup captures the amount of energy that must be invested per speedup for a given number of threads and a given frequency. Table 4 shows the EPS values for the blackscholes benchmark on Skylake, see also Fig. 5. Interestingly, the EPS is strongly related to the EDP from the last subsection due to:

$$EPS(p,s) = \frac{EDP(p,s)}{T_{seq}(s)}[J]. \tag{11}$$

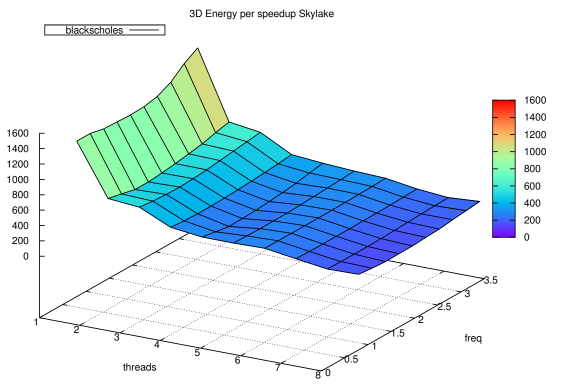
This shows that the EPS normalizes the EDP which the application-specific sequential runtime for the same frequency, and, thus, the metric EPS can be viewed as a normalized measure for energy efficiency.

A speedup metrics for the power is derived from the energy speedup and the runtime speedup by considering the power as a function $P(p,s)$ of the number of threads p and the operational frequency $f = f_{max}/s$. The *power speedup* $PS(p,s)$ is defined as energy speedup per (runtime) speedup in the following way:

$$\begin{aligned} PS(p,s) &= \tfrac{ES(p,s)}{S(p,s)} = \tfrac{E(1,s)}{E(p,s)} \cdot \tfrac{T_{par}(p,s)}{T_{seq}(s)} \\ &= \tfrac{E(1,s)}{T_{seq}(s)} \cdot \left(\tfrac{E(p,s)}{T_{par}(p,s)}\right)^{-1} \\ &= \tfrac{P(1,s)}{P(p,s)}. \end{aligned} \tag{12}$$

Table 4: EPS for the blackscholes benchmark on Skylake.

	energy per speedup			
Freq	1	2	4	8
0.8 GHz	1223.49	522.66	286.19	183.24
1.0 GHz	1227.49	527.00	274.74	172.04
1.2 GHz	1231.41	432.26	220.86	147.55
1.4 GHz	1184.56	418.35	218.99	149.59
1.5 GHz	1170.97	419.50	220.63	148.70
1.7 GHz	1176.71	406.92	226.19	153.76
1.9 GHz	1170.00	430.59	230.00	154.60
2.1 GHz	1149.51	440.77	242.19	160.82
2.3 GHz	1178.57	462.05	251.47	168.56
2.5 GHz	1211.48	479.37	263.34	177.07
2.7 GHz	1266.15	512.96	279.91	187.46
2.8 GHz	1297.01	523.02	289.31	200.30
3.0 GHz	1397.57	566.42	314.07	208.82
3.2 GHz	1472.07	602.33	334.74	224.72
3.4 GHz	1546.02	639.04	356.82	239.76



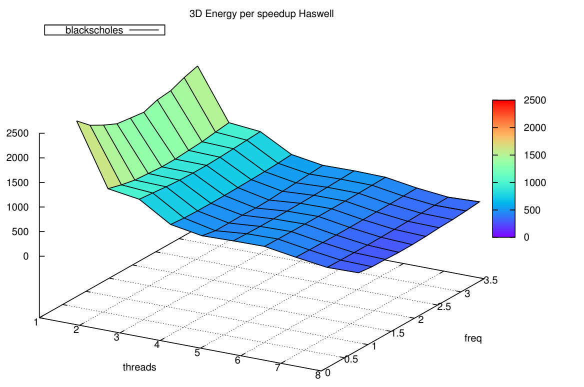


Figure 5: Energy per speedup EPS for blackscholes for different frequencies and threads on an i7 Skylake (left) and Haswell (right).

$PS(p, s)$ is a dimensionless metrics which captures the relative difference of the power consumption of an application program that occurs when using p threads instead of a sequential execution for a fixed frequency. Although PS is defined to be dependent on p and s, the calculation shows that $PS(p, s)$ is actually only dependent of p for each fixed f. Figure 6 shows the power speedups of all PARSEC benchmarks and Fig. 7 is a detailed diagram for blackscholes.

3.5 Combining power consumption and speedup

Considering the power consumption of an application program using the same operational frequency with scaling factor s, it can be observed that the power typically increases with the number of threads employed, see Table 1. This effect can be quantitatively captured with the *power increase factor* $PI(p, s)$, which we define as follows:

$$PI(p, s) = \frac{P(p, s)}{P(1, s)}, \qquad s \text{ fixed.} \tag{13}$$

Different applications lead to different values for $PI(p, s)$, which should be smaller than p in all cases. Table 5 shows PI for the blackscholes benchmark on Skylake as an example. The example shows that $PI(p, s)$ is not a linear function in p, which means that the power increase is not proportional to the number of threads employed.

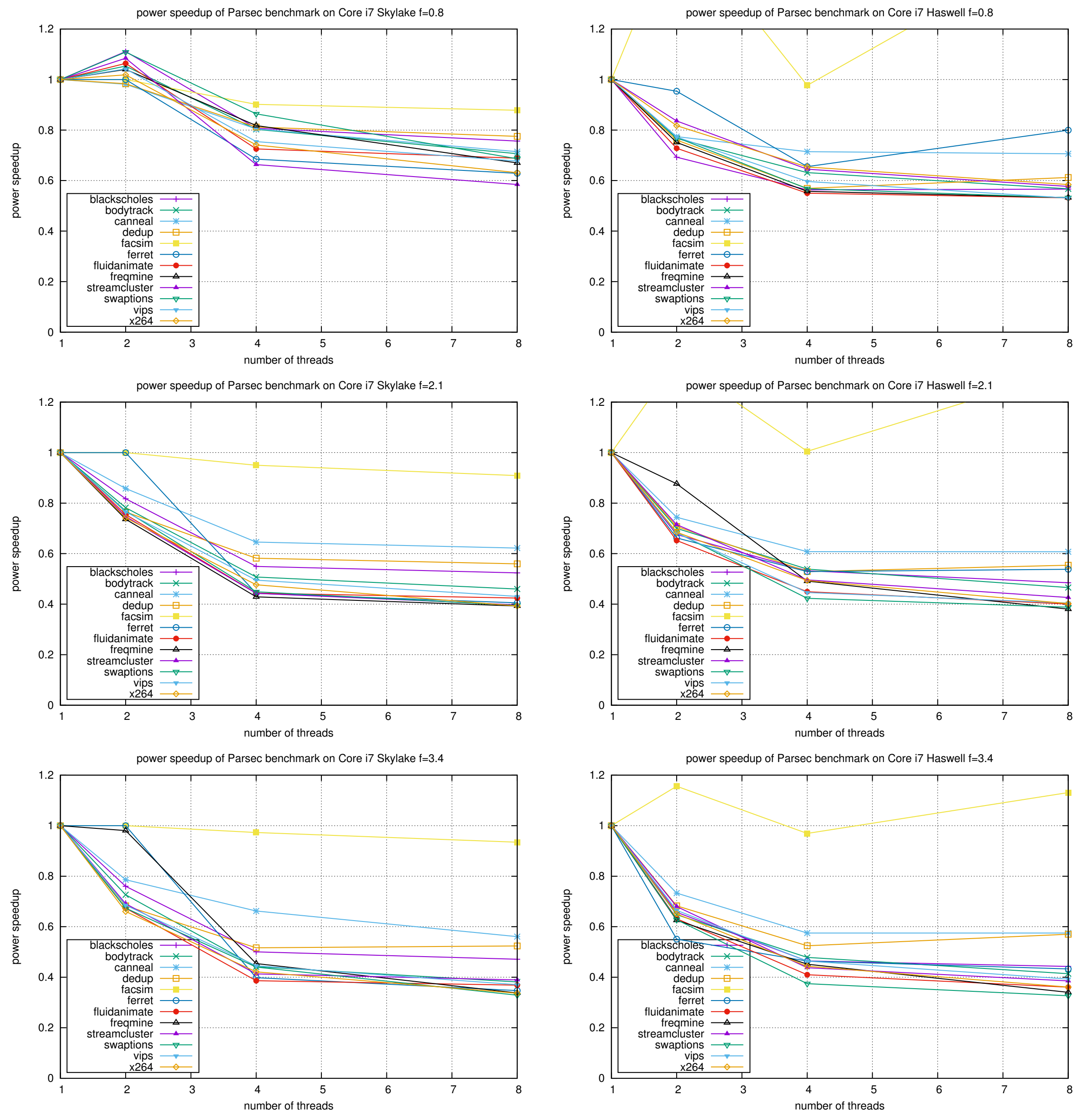
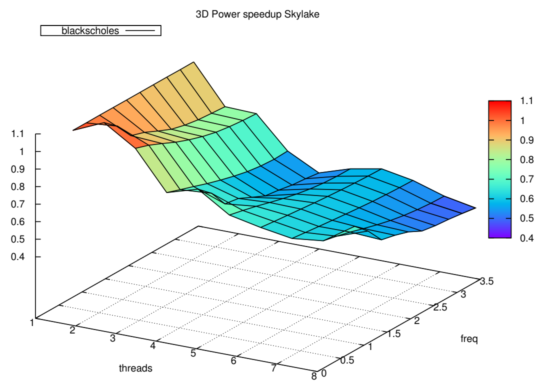


Figure 6: Power speedup PS for selected frequencies on an i7 Skylake (left) and Haswell (right): 0.8 GHz (top), 2.1 GHz (middle), and 3.4 GHz (bottom).



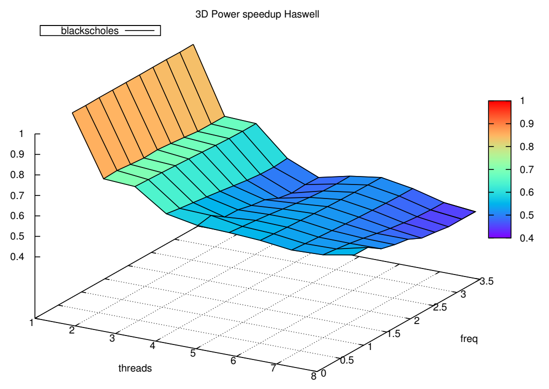
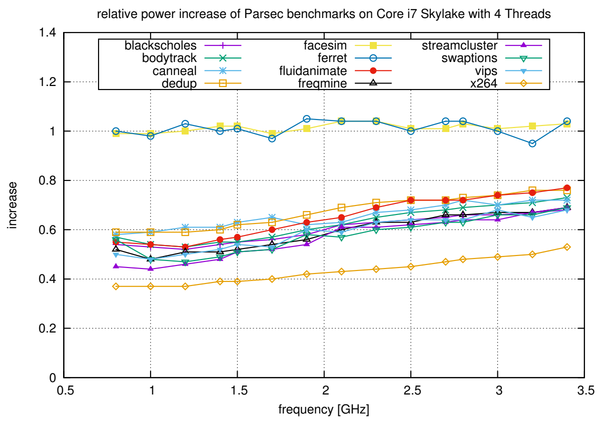


Figure 7: Power speedup PS for blackscholes for different frequencies and threads on an i7 Skylake (left) and Haswell (right).

Table 5: PI for the blackscholes benchmark on Skylake.

	power increase factor			
Freq	1	2	4	8
0.8 GHz	1.00	0.90	1.24	1.32
1.0 GHz	1.00	0.89	1.26	1.39
1.2 GHz	1.00	1.05	1.50	1.60
1.4 GHz	1.00	1.13	1.60	1.69
1.5 GHz	1.00	1.14	1.63	1.70
1.7 GHz	1.00	1.10	1.61	1.74
1.9 GHz	1.00	1.17	1.70	1.81
2.1 GHz	1.00	1.22	1.82	1.91
2.3 GHz	1.00	1.25	1.85	1.95
2.5 GHz	1.00	1.26	1.86	2.00
2.7 GHz	1.00	1.29	1.91	2.03
2.8 GHz	1.00	1.29	1.93	2.03
3.0 GHz	1.00	1.29	1.94	2.04
3.2 GHz	1.00	1.31	1.97	2.09
3.4 GHz	1.00	1.32	2.00	2.12



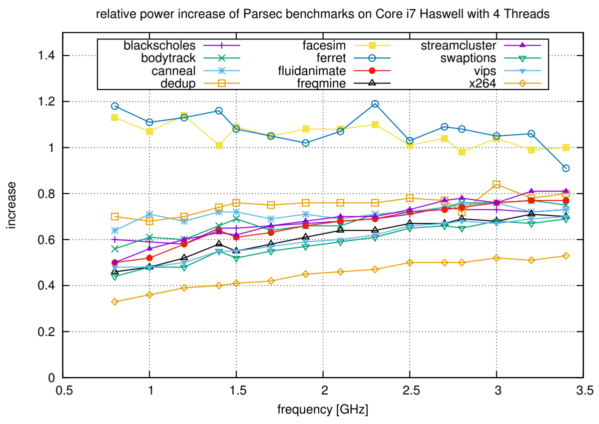


Figure 8: Relative Power Increase RPI for different frequencies using four threads on an i7 Skylake (left) and Haswell (right)

A comparison with the runtime speedup values shows a direct correlation between the runtime speedup and the power increase. Thus, the runtime speedup seems to be a good indicator for the effect, whether a significant increase of the power consumption has to be expected for a larger number of threads.

To further investigate the correlation between runtime speedup and power increase, we define the *relative power increase factor* as a quantitative measure for the mutual dependence of the runtime speedup and the power increase factor as follows:

$$RPI(p,s) = \frac{PI(p,s)}{S(p,s)} \tag{14}$$

$RPI(p,s)$ describes how much the power consumption of an application increases per runtime speedup if the number of threads is increased accordingly. If there were a linear dependence of the power consumption on the runtime speedup, we would expect an RPI value of about 1. This is indeed the case for some of the PARSEC benchmarks, but for many benchmarks, it is $RPI < 1$, i.e. the power increase is smaller than the runtime speedup. Figure 8 illustrates the RPI values for all PARSEC benchmarks using $p = 4$ threads, since the executing processors have only four physical cores.

Please note that $RPI(p,s)$ is a metric that captures effects of both independent variables p and s, and is thus able to illustrate the influence on both. Also, both the energy consumption and the parallel execution time are evaluated together. This metric is capable to reduce the influence of two independent variables onto the two dependent phenomena to one metrics. RPI has typically values between 0.5 and 1, so it is easy to read and interprete compared to other metrics such as the EDP. The RPI metrics is suitable to study and compare different applications concerning the correlation between power or energy efficiency on the one hand and effective parallelism on the other hand. A coarse evaluation criteria is that a lower RPI-value means that the inherent parallelism of an application is exploited without a significant increase of the power drawing, thus leading to a smaller overall energy consumption. Thus, a lower RPI value is better.

4 Analysis, Discussion and Summary

In this section, we evaluate the metrics from the last section for the PARSEC and SPLASH-2 benchmarks suites, discuss the results, compare the Intel Core i7 and an ARM architecture, and reflect upon usage scenarios for the metrics.

4.1 Results of the metrics applied to PARSEC and SPLASH-2

Table 6 presents measurements of some of the metrics from Sect. 3 for all PARSEC benchmarks on both the Haswell (top) and the Skylake (bottom) architecture. The table shows for each PARSEC application

(a) the minimum and maximum runtime in seconds, taken over all possible frequencies and number of threads (columns 1 and 2),

(b) the speedup obtained when using $p = 8$ threads for frequencies $f = 0.8$ GHz and $f = 3.4$ GHz (columns 3 and 4),

(c) the minimum and maximum energy consumption in Joule, taken over all possible frequencies and number of threads (columns 5 and 6),

(d) the operational frequency for which the minimum energy consumption results for the cases of one and eight threads (columns 7 and 8),

(e) the energy speedup ES using $p = 8$ threads for the smallest and the largest frequency available (columns 9 and 10),

(f) the minimum and maximum energy per speedup EPS in Joule, taken over all possible frequencies and number of threads (columns 11 and 12), and

(g) the minimum and maximum relative power increase factor RPI when using four threads, taken over all frequencies (columns 13 and 14).

Table 7 shows the same information for the SPLASH-2 benchmarks.

Usually, it is not needed to consider all metrics. Rather, it is reasonable to clearly specify the specific goal for an appraisal of an application program, a set of application programs, or different implementations of the same application. The selection of metrics presented in Table VI and VII has been guided by a specific goal to assess the possibility of energy minimization of a set of benchmarks when the frequency and the number of threads can be freely chosen. Clearly for this goal, the interval boundaries (minimum and maximum values) of the possible runtimes and the possible energy consumptions for all possible frequencies and numbers of threads available belong to the core information illustrating a possible need or potential for energy minimization. The frequencies (columns 7 and 8 in Table VI and VII) leading to the minimal energy is also a first indicator for the potential of optimization. The last three sets of columns (column 9 to 14) are devoted to the metrics chosen for this appraisal-criteria: The energy speedup, the energy per speedup and the relative power increase factor are all metrics evaluating the energy with respect to performance and speedup. And thus they are possible metrics for an estimate of the energy efficiency. The specific results of the appraisal for the PARSEC and SPLASH-2 benchmarks is given in the next subsection.

4.2 Interpretation of the results

For both the PARSEC and the SPLASH-2 applications, it can be observed that there is a large variation of the runtime (columns 1 and 2) and the energy consumption (columns 5 and 6) when different frequencies or number of threads are used, i.e., the specific selection of the parameters has a large impact and a significant amount of energy can be saved if the right combination is used. For the Haswell, the frequency for which the minimum energy consumption results lies between 1.2 GHz and 2.3 GHz for the PARSEC benchmarks and between 0.8 GHz and 3.2 GHz for the SPLASH-2 benchmarks, with smaller frequencies for a larger number of threads, see columns 7 and 8. Thus, the SPLASH-2 benchmarks show a larger variation. For the Skylake, the corresponding frequencies lie between 0.8 GHz and 2.1 GHz for the PARSEC benchmarks and between 0.8 GHz and 1.9 GHz, again with smaller frequencies for a larger number of threads. For this processor, no significant difference between the PARSEC and the SPLASH-2 applications results.

For most of the applications, a significant runtime speedup can be obtained by executing them with multiple threads. The speedup values obtained differ slightly when different operational frequencies are used, see columns

Table 6: Evaluation of different Parsec Benchmarks on Haswell (first half) and Skylake (second half) using runtime in seconds, engery in joule and Power in Watt

	Runtime		**Speedup**		**Energy Consumed**		**Best Freq**		**ES (p=8)**		**EPS**		**RPI (p=4)**	
Benchmark	min	max	f=0.8	f=3.4	min	max	p=1	p=8	f=0.8	f=3.4	min	max	min	max
blackscholes	37.12	558.67	3.60	3.54	948.56	2323.36	2.3	1.2	2.04	1.57	260.22	2323.36	0.60	0.73
bodytrack	31.52	463.37	3.72	3.47	868.69	2134.53	2.3	1.2	2.11	1.44	234.26	2134.53	0.56	0.77
canneal	71.00	527.51	2.59	2.66	1209.09	2851.98	1.5	1.2	1.83	1.53	460.53	2851.98	0.64	0.76
dedup	6.25	51.30	2.77	2.09	182.23	334.58	2.3	2.1	1.69	1.19	68.04	397.87	0.68	0.84
facesim	0.27	1.11	1.00	0.93	3.84	6.07	2.3	2.3	0.88	0.95	3.84	6.12	0.98	1.14
ferret	0.26	1.07	1.00	0.94	3.63	5.89	2.3	2.3	0.86	1.07	3.63	6.00	0.91	1.19
fluidanimate	63.30	1054.09	4.32	4.02	1949.83	5077.73	2.3	1.2	2.30	1.45	453.43	5077.73	0.50	0.77
freqmine	80.63	1581.32	4.67	4.65	2742.39	7700.87	2.1	1.2	2.49	1.58	586.20	7700.87	0.46	0.71
streamcluster	71.25	918.56	5.19	4.99	1564.56	5222.48	1.7	1.2	3.00	1.93	315.75	5442.64	0.50	0.81
swaptions	39.46	867.48	5.18	5.18	1411.24	4313.39	2.3	1.2	2.76	1.69	269.41	4313.39	0.44	0.69
vips	21.63	341.41	4.30	3.76	648.34	1703.58	2.1	1.2	2.28	1.48	151.61	1703.58	0.48	0.70
x264	16.87	396.76	5.95	5.75	560.11	2155.81	2.1	1.2	3.48	2.08	94.92	2155.81	0.33	0.53
blackscholes	34.02	333.86	2.97	3.70	539.53	1546.02	2.1	1.2	2.25	1.74	147.55	1546.02	0.52	0.68
bodytrack	27.15	294.10	2.91	3.91	490.99	1314.92	1.7	1.2	2.06	1.49	126.06	1314.92	0.53	0.73
canneal	67.94	456.88	2.80	2.77	650.01	2015.27	1.5	0.8	2.00	1.55	232.10	2015.27	0.58	0.72
dedup	5.38	43.14	2.36	2.65	84.86	207.75	1.7	1.0	1.83	1.39	30.00	207.75	0.59	0.76
facesim	0.26	0.99	1.01	1.00	2.33	3.47	1.7	1.7	0.98	0.93	2.32	3.75	0.99	1.03
ferret	0.49	1.72	1.03	0.99	4.53	7.17	2.1	1.5	0.99	0.97	4.09	8.28	0.95	1.05
fluidanimate	60.51	636.68	2.97	3.95	1151.72	3151.76	1.9	1.0	2.04	1.46	282.69	3151.76	0.53	0.77
freqmine	75.31	1011.15	3.55	4.83	1543.42	4726.40	2.1	1.0	2.38	1.63	322.56	4726.40	0.48	0.69
streamcluster	70.36	878.20	5.29	4.77	878.56	3851.71	1.5	0.8	3.09	1.86	144.19	3851.71	0.44	0.69
swaptions	35.91	516.00	3.68	5.31	779.57	2529.71	1.9	1.0	2.53	1.75	149.48	2529.71	0.47	0.69
vips	18.54	221.89	3.28	4.27	367.38	1067.31	2.1	1.0	2.22	1.58	84.57	1067.31	0.48	0.68
x264	14.61	255.89	4.92	6.20	304.57	1197.61	1.7	1.0	3.11	2.09	48.37	1197.61	0.37	0.53

Table 7: Evaluation of the metrics for the Splash2 Benchmarks on Haswell (first half) and Skylake (second half)

	Runtime		**Speedup**		**Energy Consumed**		**Best Freq**		**ES (p=8)**		**EPS**		**RPI (p=4)**	
Benchmark	min	max	f=0.8	f=3.4	min	max	p=1	p=8	f=0.8	f=3.4	min	max	min	max
barnes	25.29	444.98	4.34	4.21	1067.37	2477.43	2.30	2.10	1.99	1.61	251.33	2477.43	0.51	0.72
cholesky	0.35	1.51	1.12	1.13	6.18	9.68	2.30	3.00	0.90	0.92	5.71	8.68	1.00	1.05
fft	148.98	595.58	2.66	2.29	1809.84	4830.14	0.80	0.80	1.82	1.54	373.47	4830.14	0.69	0.83
fmm	24.75	317.68	3.54	2.52	831.06	1670.93	2.30	1.40	1.76	1.32	241.21	1670.93	0.55	0.74
lu_cb	26.32	408.42	3.61	3.61	1061.13	2254.42	3.00	1.70	1.65	1.28	286.33	2254.42	0.56	0.74
lu_ncb	34.70	502.74	3.88	3.69	1407.10	2693.82	2.10	1.90	1.63	1.27	375.23	2675.30	0.70	1.09
ocean_cp	49.72	289.87	3.64	1.51	953.77	2106.09	2.30	1.00	1.98	0.66	269.88	1943.71	0.51	1.43
ocean_ncp	54.00	537.91	5.62	2.50	1113.04	3426.67	2.70	1.00	2.92	1.12	208.33	3426.67	0.48	1.02
radiosity	102.44	534.97	1.12	1.16	2065.73	3002.57	3.00	2.10	1.00	0.91	395.28	3002.57	0.95	1.17
radix	7.94	139.99	6.50	4.21	203.03	779.66	3.20	1.70	3.47	1.75	32.87	779.66	0.48	0.70
raytrace	22.90	489.52	5.07	5.01	1158.39	2938.06	2.30	1.40	2.33	1.66	229.44	2938.06	0.52	0.71
volrend	24.73	480.42	4.55	4.55	1031.52	2826.36	2.30	1.70	2.37	1.66	224.74	2826.36	0.50	0.70
water_nsquared	82.97	1252.57	3.77	3.64	3604.37	8221.01	2.30	1.90	1.91	1.26	446.20	8221.01	0.55	0.88
water_spatial	24.75	490.37	4.72	4.85	992.49	2889.17	3.20	1.70	2.53	1.79	208.22	2889.17	0.46	0.64
barnes	21.39	232.84	3.05	4.48	417.23	1244.07	1.7	1.2	1.55	1.61	100.79	1244.07	0.52	0.68
cholesky	0.32	1.32	1.10	1.13	3.39	5.14	1.7	1.5	0.96	0.96	3.10	4.95	0.99	1.02
fft	23.55	90.83	1.50	1.65	258.05	580.93	1.5	1.5	1.23	1.15	163.02	580.93	0.78	0.87
fmm	18.64	198.87	3.21	3.88	322.22	926.49	1.7	1.0	2.09	1.50	81.17	926.49	0.53	0.77
lu_cb	28.65	317.63	2.97	4.24	576.70	1573.95	1.5	1.2	2.03	1.50	137.53	1573.95	0.51	0.70
lu_ncb	34.78	407.64	3.36	4.18	652.01	1892.70	1.5	1.0	2.11	1.45	145.82	1892.70	0.53	0.96
ocean_cp	34.98	126.45	2.36	1.59	335.16	784.12	1.5	0.8	1.59	0.82	125.37	784.12	0.66	1.19
ocean_ncp	39.60	139.62	2.24	1.58	388.21	1103.04	0.8	1.0	1.45	0.83	146.66	907.70	0.71	1.22
radiosity	106.78	317.35	1.13	1.21	1073.33	1740.68	1.7	1.4	1.00	0.85	943.07	1559.80	0.99	1.11
radix	6.48	94.27	4.77	5.52	78.92	346.79	1.9	1.4	3.63	2.34	11.90	346.79	0.42	0.61
raytrace	20.88	273.69	3.46	4.98	477.11	1498.99	1.7	1.0	2.24	1.67	97.71	1498.99	0.52	0.70
volrend	23.71	298.27	3.33	4.80	480.75	1437.68	1.7	1.0	2.23	1.61	102.06	1437.68	0.51	0.69
water_nsquared	72.83	715.78	2.91	3.84	1519.57	4015.32	1.5	1.0	1.85	1.31	394.11	4015.32	0.59	0.85
water_spatial	22.35	279.36	3.56	4.80	439.87	1376.12	1.7	1.0	2.29	1.68	89.69	1376.12	0.47	0.63

3 and 4 in Tables 6 and 7, and depend on the specific benchmark. For the Skylake architecture, using a larger frequency leads to a larger speedup for most of the applications. This behavior is slightly stronger for the SPLASH-2 applications than for the PARSEC applications. Such an increase of the speedup with the operational frequency cannot be observed for the Haswell architecture. Instead, many applications even exhibit a descrease of the speedup when using a larger operational frequency.

Considering the energy speedup for the different applications in columns 9 and 10 of Tables 6 and 7, it can be seen that the resulting values are larger than 1 for most of the applications. This corresponds to the observation that the energy consumption decreases when using a larger number of threads. Therefore, it is expected that $ES(p, s) > 1$. This is confirmed for most of the PARSEC applications, as can be seen in column 5 of Table 6. There are a few exceptions, which are caused by applications with a very small energy consumption and no significant runtime speedup such as facesim or ferret from PARSEC or cholesky from SPLASH-2. For most of the applications, the energy speedup decreases with the operational frequency, which means that for larger frequencies the energy saving obtained when using more threads is not as significant as for smaller frequencies. There is no significant difference between the PARSEC and the SPLASH-2 applications, and also not between the Haswell and Skylake architecture.

The RPI values in the last columns show a surprising match of the values on the two processors. This suggests that the RPI expresses the essence of the application behavior. Only the applications with a very low execution time have an RPI larger than 1. The other applications have RPI values of about 0.5. There is one exception which is application x264 with a minimum RPI value of 0.33 and 0.37, respectively. The same application also achieves the highest speedup.

4.3 Comparison of different architectures

When comparing the Intel Core i7 architectures Haswell and Skylake, it can be observed that the Skylake typically leads to smaller execution times. This can be seen at the minimum and maximum execution times in Table 6, but also when considering the execution times for the whole range of frequencies and number of threads, which are not shown in a figure. Even more significant is the fact that the Skylake processor needs much less energy for executing an application than the Haswell processor, indicating a more energy-efficient architectural design. The best frequencies for the energy consumption are typically smaller for Skylake than for Haswell, which can be explained by the power characteristics shown in Figs. 1 and 2: the Skylake shows a larger decrease of the power consumption than the Haswell architecture when the operational frequency is decreased starting with the maximum frequency, i.e., a reduction of the frequency has a larger impact for the Skylake than for the Haswell architecture, and this effect influences the energy consumption correspondingly. The smallest energy consumptions typically result for frequencies that are larger than the minimum frequency provided.

In order to study whether the effects observed are specific for the Intel Core i7 architectures, we have tested some of the PARSEC applications for an ARM architecture. More precisely, we have considered the Odroid XU4 with the ARM big.LITTLE architecture, providing DVFS and allowing a frequency variation between 0.2 GHz and 2.0 GHz for the Cortex A15 and between 0.2 GHz and 1.4 GHz for the Cortex A7, both with 0.1 GHz steps. The energy and power measurements have been performed using a second board with analogue digital converter provided. In the following, we investigate the PARSEC application blackscholes as example, which has already been used in the previous section as running example. The following observations can be made:

- The power consumption increases with the operational frequency. However, compared to the Intel Core i7 systems, this increase is less steep. As example, Table 8 shows the power consumption for the blackscholes benchmark for different frequencies and different numbers of threads. It can also be observed that the overall power consumption is much smaller than on the Core i7 systems. For the Odroid XU4, the power increase factor PI increases only slightly with the number of threads, see Table 10, and it is smaller than the corresponding PI values for the Core i7 systems, see Table 5 for the Skylake.

	power consumption threads			
Freq	1	2	4	8
0.2	2.44	3.06	2.52	2.60
0.3	2.51	2.61	2.62	2.71
0.4	2.57	2.70	2.66	2.81
0.5	2.63	2.80	2.80	2.93
0.6	2.69	2.91	2.89	3.07
0.7	2.75	3.00	2.98	3.17
0.8	2.84	3.13	3.12	3.34
0.9	2.96	3.27	3.25	3.53
1.0	3.06	3.41	3.30	3.71
1.1	3.17	3.58	3.54	4.03
1.2	3.27	3.75	3.75	4.15
1.3	3.40	3.93	3.95	4.40
1.4	3.52	4.10	4.10	4.63
1.5	3.63	4.29	4.23	4.81
1.6	3.81	4.59	4.17	5.06
1.7	3.99	4.87	4.35	5.33
1.8	4.23	5.21	4.50	5.62
1.9	4.51	5.70	5.17	6.09
2.0	4.93	6.41	5.12	6.54

Table 8: Power Consumption in Watt of the Blackscholes benchmark on Odroid XU4 for different frequencies and different number of threads

	energy per speedup			
Freq	1	2	4	8
0.2	11561.67	4665.48	2788.56	1175.03
0.3	7844.16	2629.16	1940.31	828.50
0.4	6028.86	2042.55	1484.84	627.25
0.5	4938.38	1680.23	1262.86	525.31
0.6	4240.47	1468.31	1085.29	467.08
0.7	3706.72	1292.05	970.65	415.42
0.8	3363.54	1188.27	885.25	385.86
0.9	3125.81	1100.47	825.38	365.54
1.0	2887.67	1042.74	770.40	353.00
1.1	2749.21	1004.97	669.07	322.09
1.2	2602.31	961.47	728.44	336.13
1.3	2496.77	951.37	719.34	329.37
1.4	2413.06	910.50	698.76	327.92
1.5	2320.11	892.90	750.75	344.11
1.6	2292.03	901.43	758.42	368.54
1.7	2259.22	893.10	822.14	392.32
1.8	2270.24	908.24	874.67	424.81
1.9	2300.43	954.72	1036.87	471.12
2.0	2376.84	1021.98	1065.69	518.81

Table 9: Energy per Speedup for the Blackscholes benchmark on Odroid XU4 for different frequencies and different number of threads

- The runtime on the Odroid XU4 is much larger than on the Core i7 systems for all frequencies. Correspondingly, the overall energy consumption is also larger than on the Core i7 systems (not shown in a figure). This has also an influence on the EDP and the energy per speedup EPS, which are both much larger on the Odroid XU4 architecture than on the Core i7 systems, see Table 9 for the EPS on the Odroid XU4 and Table 4 for the EPS on the Syklake system.
- The runtime reduction factors are quite similar on both systems, if the same scaling factors are used. Larger deviations occur if the number of threads increases.
- The behavior of the relative power increase factors RPI for the Odroid XU4 is quite similar than for the Core i7 systems, see Table 11. The RPI values get smaller if the operational frequency is descreased and if the number of threads is increased.

These observations suggest that the new metrics introduced are well suited to evaluate the execution of multithreaded applications on different architectures, to study their behavior for frequency scaling and parallelism, and to capture fundamental differences of the applications' energy sensitivity.

4.4 How and when to use the metrics

In this article, we have used or defined several metrics, which are T, E, P, S, R, ES, ER, EDP, EPS, PS, PI, RPI. In the following, we briefly describe how and when to use them.

The runtime T, energy consumption E and power consumption P of an application can be measured for different operational frequencies and different numbers of threads and can be used for a comparison of different implementation variants of the same application. But these data do not explicitly provide an information about the energy or power sensitivity of an application. The (runtime) speedup S is the traditional metric to investigate the runtime scalability of a parallel application. However, this metric does not take frequency scaling into account. The runtime reduction factor R is a modification of the runtime speedup that emphasises on the change in the runtime when

	power increase factor			
Freq	1	2	4	8
0.2	1.00	1.25	1.03	1.06
0.3	1.00	1.04	1.04	1.08
0.4	1.00	1.05	1.04	1.10
0.5	1.00	1.07	1.07	1.11
0.6	1.00	1.08	1.08	1.14
0.7	1.00	1.09	1.09	1.15
0.8	1.00	1.10	1.10	1.18
0.9	1.00	1.11	1.10	1.19
1.0	1.00	1.11	1.08	1.21
1.1	1.00	1.13	1.11	1.27
1.2	1.00	1.15	1.15	1.27
1.3	1.00	1.16	1.16	1.29
1.4	1.00	1.17	1.17	1.32
1.5	1.00	1.18	1.17	1.33
1.6	1.00	1.21	1.10	1.33
1.7	1.00	1.22	1.09	1.34
1.8	1.00	1.23	1.06	1.33
1.9	1.00	1.26	1.14	1.35
2.0	1.00	1.30	1.04	1.33

Table 10: Power Increase Factor for the Blackscholes benchmark on Odroid XU4 for different frequencies and different number of threads

	RPI threads			
Freq	1	2	4	8
0.2	1.00	0.71	0.50	0.33
0.3	1.00	0.59	0.51	0.34
0.4	1.00	0.60	0.51	0.34
0.5	1.00	0.60	0.52	0.34
0.6	1.00	0.61	0.52	0.35
0.7	1.00	0.62	0.53	0.36
0.8	1.00	0.62	0.54	0.37
0.9	1.00	0.62	0.54	0.37
1.0	1.00	0.63	0.54	0.39
1.1	1.00	0.64	0.52	0.39
1.2	1.00	0.65	0.57	0.40
1.3	1.00	0.66	0.58	0.41
1.4	1.00	0.66	0.58	0.42
1.5	1.00	0.67	0.61	0.44
1.6	1.00	0.69	0.60	0.46
1.7	1.00	0.69	0.63	0.48
1.8	1.00	0.70	0.64	0.50
1.9	1.00	0.72	0.72	0.53
2.0	1.00	0.75	0.68	0.54

Table 11: Relative Power Increase of the Blackscholes benchmark on Odroid XU4 for different frequencies and different number of threads

the operational frequency is changed for executing an application using the same number of threads. Thus, R can be used to explore the DVFS behavior of applications and the influence of frequency changes on the runtime. The energy reduction factor ER has the same flavor as the runtime reduction factor R, but considers the influence of DVFS on the energy consumption instead of the runtime.

The energy sensitivity or energy efficiency can be assessed with the metrics ES, EDP, and EPS. The energy speedup ES can be considered as an energy-oriented modification of the runtime speedup, as it investigates the energy consumption of an application for a varying number of threads using a fixed operational frequency. In particular, ES evaluates the energy consumption when the number of threads is increased for the execution of a multithreaded application. A value $ES(p,s) < 1$ would indicate that using more threads would lead to a higher energy consumption than a sequential execution, whereas $ES(p,s) > 1$ means that using more threads leads to a reduction of the energy consumption. Tables 6 and 7 show that nearly all PARSEC and SPLASH-2 applications have ES values larger than 1, i.e., a parallel execution is beneficial for the energy consumption. The metrics can be used to compare the energy sensitivity of different applications or of different implementations of the same applications: an implementation with a higher value of $ES(p,s)$ leads to more energy saving than an implementation with a smaller ES value for the same number of threads and is therefore preferable if the energy consumption should be kept small. The metric energy per speedup EPS is strongly related to the traditional EDP metric, see Equ. (11). The normalization with the sequential runtime allows a comparison of different applications or implementations of the same application, which is not staightforwardly possible with the EDP metric. When comparing different implementations, smaller EPS values indicate a more efficient energy utilization for a parallel execution with a certain number of threads.

The power behavior with respect to varying frequencies and numbers of threads is in the focus of the metrics PS, PI, and RPI. The power speedup PS investigates the development of the power consumption of an application when the number of executing threads is increased. Typically, the power consumption increases with the number of threads and PS captures the increase factor. The metric can be used to compare the power sensitivity of different applications when the number of threads is increased using a fixed operational frequency. The power increase factor PI is the inverse of the power speedup, which is exploited for the next metric. The relative power increase factor RPI sets the power increase and the runtime speedup in relation. The metric can be used to investigate whether the increase of the power consumption of an application is related to a corresponding increase in the runtime speedup, or in other words, whether the additional power invested when using more threads is rewarded by a corresponding decrease in the runtime. A small RPI value indicates a good and efficient power usage. Thus, when comparing different implementation variants of an application, a variant with a smaller RPI value should be preferred.

The metrics introduced can be used to analyze the runtime and energy behavior of multithreaded applications of DVFS architectures using different number of threads for their execution. Future work might also address the usage of the metrics for system software to control the DVFS setting, e.g., by an inclusion in a new governor. Especially the novel metrics relative power increase factor RPI is capable to express all runtime and energy effects and integrate them in a single measure. The metrics RPI has also the advantage that due to its scale between about 0.5 and 1, the application-specific values are easy to grasp and interprete by the application programmer. Not only the absolute values but also significant deviations or anomalies are quickly detected. Thus, the RPI metrics is a promising and easy-to-use metrics for a combined assessment of energy consumption and execution time depending on both variables.

5 Related Work

Power saving is in the focus of research for some years now and several features have been integrated in computer systems of different sizes and classes, from handheld devices to large servers [28, 12, 13]. A well studied and accepted technique is DVFS, which allows the hardware to lower the operational voltage and frequency at the cost of a possibly higher execution time, as documented in [33]. The approach to determine the voltage scaling factor that minimizes the total CPU energy consumption by taking both the dynamic power and the leakage power into consideration has been discussed in [19, 18, 33] for sequential executions. Power models for multicore designs are developed in [12]. A detailed analytical treatment of energy models in the context of scheduling algorithms is given in [20]. The energy model used in this article has been used in [23] to analyse scheduling algorithms for fork-join parallelism. Fundamental aspects of speed scaling and power-down scheduling are described in [11].

The energy delay product (EDP) has first been introduced in [15] to capture both energy consumption and execution time simultaneously. However, as it has been pointed out in [1], the EDP is not able to distinguish whether a smaller execution time or a better energy utilization leads to a smaller EDP value when comparing different version of a program. The biased effects of the EDP and its generalization ED^n for a nonnegative integer n are also discussed in [16]. To separate energy and performance when considering different program versions, [1] introduces powerup and greenup metrics, which are similar to our power speedup and energy speedup metrics. These metrics are then used together with the speedup to identify different energy categories of software. In our article, we go into a different direction by investigating the relationship between speedup, energy, and power. In the context of frequency scaling, [14] defines the power-aware speedup, which incorporates the frequency when computing the runtime speedup of programs. However, the energy consumption is not explicitly taken into consideration. In the context of task scheduling policies, [21] has proposed the metrics speedup per Watt (SPW), power per speedup (PPS), and energy per target (EPT) and has evaluated task scheduling policies using these metrics.

The investigation of multithreaded application programs is an ever-present research topic for several years now. Especially, the properties and the behavior of the PARSEC and the SPLASH-2 benchmark suites have been studied intensively with repect to different criteria, for which we give some representative examples. In [5], the motivation and design goals of PARSEC have been documented. The SPLASH-2 benchmark has been introduced in [32]. A first qualitative comparison of both benchmarks is given in [4]. In [2], the comparison concentrates on the communication characteristics of the suites. More recently, the workload scalability of the PARSEC has been analysed in [29]. A system to dynamically adapt a multithreaded programs' parallelism has been proposed in [30]; it was applied to several benchmarks including some from the PARSEC. Vectorization approaches for PARSEC benchmarks have also been considered in [9] and the energy consumption has been reported. The performance and energy impact for parallelization and vectorization has been studied for the PARSEC benchmarks in [8] for the Intel core i5 and i7 with up to 8 threads and for an ARM architecture for up to 32 threads. The above mentioned studies for PARSEC and/or SPLASH-2 have mainly used the execution time, the speedup or the energy consumption for measuring the performance but have ignored other measures or fused metrics to appraise the results as it was done in our work. Also, a joint study and comparison of parallelism and frequency scaling has been missing so far in the investigations of PARSEC and SPLASH-2 and our work is contribution in that direction.

A detailed energy roofline model, which also takes details of the processor architecture such as memory hierarchies into consideration, has been presented in [10]. The model has been evaluated with microbenchmarks for different architectures including x86, ARM, and GPU.

6 Conclusions

The growing interest in energy efficient computing leads to a demand for appropriate ways to appraise the behavior of program execution. Especially for parallel executions on DVFS processors, the mutual interaction and contrary effects of the parallelism exploited and the energy consumed are quite complex. To assess the program behavior, we have proposed and evaluated several metrics, including energy speedup, energy reduction factor, energy per speedup, power speedup, power increase factor, and relative power increase factor, which capture the effects of a varying number of threads and frequencies on performance and energy in isolation or in a combined measure.

The metrics have been used to assess the multithreaded applications of the PARSEC benchmark suite with the goals to illustrate the expressiveness of the metrics and to appraise the energy behavior of the benchmarks. Furthermore, the experiments have been repeated for the SPLASH-2 benchmark suite to have a broader experimental foundation of the results. A detailed analysis of the metrics discusses the usage and usage scenarios of the metrics explaining how and when to use a specific metric or set of metrics for appraising a specific optimization goals.

Acknowledgement

This work was partially performed within the Federal Cluster of Excellence EXC 1075 ”MERGE Technologies for Multifunctional Lightweight Structures”, supported by the German Research Foundation (DFG). This work is also supported by the German Ministry of Science and Education (BMBF), project SeASiTe, number 01IH16012A/B.

References

[1] S. Abdulsalam, Z. Zong, Q. Gu, and Meikang Qiu. Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency. In *Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International*, pages 1–8, Dec 2015.

[2] N. Barrow-Williams, C. Fensch, and S. Moore. A Communication Characterisation of Splash-2 and Parsec. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC ’09, pages 86–97. IEEE Computer Society, 2009.

[3] C. Bienia, S. Kumar, and K. Li. PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors. In *Proc. of the 2008 Int. Symp. on Workload Characterization*, September 2008.

[4] C. Bienia, S. Kumar, and Kai Li. PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors. In *Proc. of the 2008 IEEE International Symposium on Workload Characterization*, pages 47–56, Sept 2008.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT ’08, pages 72–81. ACM, 2008.

[6] C. Bienia, S. Kumar, J.P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proc. of the 17th Int. Conf. on Parallel Architectures and Compilation Techniques*, October 2008.

[7] J.A. Butts and G.S. Sohi. A static power model for architects. In *In Proc. of the 33rd Int. Symp. on Microarchitecture (MICRO-33)*, 2000.

[8] J. M. Cebrián, L. Natvig, and J. C. Meyer. Performance and energy impact of parallelization and vectorization techniques in modern microprocessors. *Computing*, 96(12):1179–1193, 2014.

[9] J. M. Cebrián, M. Jahre, and L. Natvig. Optimized hardware for suboptimal software: The case for SIMD-aware benchmarks. In *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 66–75, March 2014.

[10] J. Choi, M. Dukhan, X. Liu, and R. Vuduc. Algorithmic Time, Energy, and Power on Candidate HPC Compute Building Blocks. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 447–457, May 2014.

[11] M. Chrobak. Algorithmic Aspects of Energy-Efficient Computing. In I. Ahmad and S. Ranka, editors, *Handbook of Energy-Aware and Green Computing*, pages 311–329. CRC Press, 2012.

[12] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *Commun. ACM*, 56(2):93–102, February 2013.

[13] A. Fedorova, J.C. Saez, D. Shelepov, and M. Prietol. Maximizing Power Efficiency with Asymmetric Multicore Systems. *Commun. ACM*, 52(12):48–57, December 2009.

[14] R. Ge and K. W. Cameron. Power-Aware Speedup. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10, March 2007.

[15] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, Sep 1996.

[16] C. H. Hsu, W. C. Feng, and J. S. Archuleta. Towards efficient supercomputing: a quest for the right metric. In *19th IEEE Int. Parallel and Distributed Proc. Symp.*, pages 8 pp.–, April 2005.

[17] Intel. *Intel 64 and IA-32 Architecture Software Developer's Manual, System Programming Guide*, 2011.

[18] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[19] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 275–280. ACM, 2004.

[20] Keqin Li. Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. *IEEE Trans. Parallel Distrib. Syst.*, 19(11):1484–1497, 2008.

[21] J. Mair, K. Leung, and Z. Huang. Metrics and task scheduling policies for energy saving in multicore computers. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 266–273, Oct 2010.

[22] T. Rauber and G. Rünger. *Parallel Programming for Multicore and Cluster Systems*. Springer Verlag, 2013.

[23] T. Rauber and G. Rünger. Modeling and Analyzing the Energy Consumption of Fork-Join-based Task Parallel Programs. *Concurrency and Computation: Practice and Experience*, 27(1):211–236, 2015.

[24] T. Rauber, G. Rünger, and M. Schwind. Energy Measurement and Prediction for Multi-Threaded Programs. In *22nd High Performance Computing Symposium 2014 (HPC 2014), Part of the SCS Spring Simulation Conference*, 2014.

[25] T. Rauber, G. Rünger, M. Schwind, H. Xu, and S. Melzner. Energy Measurement, Modeling, and Prediction for Processors with Frequency Scaling. *The Journal of Supercomputing*, 2014.

[26] G. Rong, F. Xizhou, S. Shuaiwen, C. Hung-Ching, L. Dong, and K.W. Cameron. PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):658 –671, may 2010.

[27] E. Rotem, A. Naveh, A. Ananthakrishnan, D. Rajwan, and E. Weissmann. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro*, 32(2):20–27, 2012.

[28] E. Saxe. Power-efficient software. *Commun. ACM*, 53(2):44–48, 2010.

[29] G. Southern and J. Renau. Analysis of PARSEC workload scalability. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 133–142, April 2016.

[30] S. Sridharan, G. Gupta, and G. S. Sohi. Adaptive, Efficient, Parallel Execution of Parallel Programs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 169–180. ACM, 2014.

[31] J. Treibig, G. Hager, and G. Wellein. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In *39th International Conference on Parallel Processing Workshops*, ICPP '10, pages 207–216. IEEE Computer Society, 2010.

[32] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22Nd Annual International Symposium on Computer Architecture*, ISCA '95, pages 24–36. ACM, 1995.

[33] J. Zhuo and C. Chakrabarti. Energy-efficient dynamic task scheduling algorithms for DVS systems. *ACM Trans. Embed. Comput. Syst.*, 7(2):1–25, 2008.