# Impact of Thread and Frequency Scaling on Performance and Energy Efficiency: An Evaluation of Core i7-8700K Using SPEC CPU2017

Ranjan Hebbar S R
*Electrical and Computer Engineering,*
*The University of Alabama in Huntsville;*
Huntsville, AL, USA
rr0062@uah.edu

Aleksandar Milenković
*Electrical and Computer Engineering,*
*The University of Alabama in Huntsville*
Huntsville, AL, USA
milenka@uah.edu

***Abstract*— This paper describes the results of measurement-based studies focusing on performance and energy efficiency of SPEC CPU2017 *speed* and *rate* benchmark runs on the Intel's Core i7-8700K processor. We measure execution time and the total energy consumed during individual benchmark runs with 1 thread/copy (1T/1C) and 6 threads/copies (6T/6C) when the processor clock is set to its nominal frequency of 3.7 GHz. Next, we evaluate how changes in the processor clock frequency impact performance, energy efficiency, as well as a composite metric called *PE* that captures both performance and energy efficiency. We find that 1T/1C benchmark runs at the nominal frequency achieve an optimal *PE*, whereas 6T/6C benchmark runs achieve optimal *PE* when running at 2.7 GHz.**

***Keywords— Performance Evaluations, Benchmarks, Energy Efficiency, Frequency Scaling, SPEC CPU2017***

## I. INTRODUCTION

In the last few decades the demand for high performance computing and data center capacity have grown exponentially. Datacenters and cloud computing act as a backbone of the IT infrastructure. Data centers in the US alone consume about 2% of the country's total energy consumption [1]. With most of the data centers still relying on conventional energy grids, it is vital to optimize energy consumption which will reduce operating temperature and improve efficiency.

Modern Intel processors, starting with the Sandy Bridge microarchitecture, have the Running Average Power Limit (RAPL) interface [2] that is designed to limit on-chip power usage while ensuring maximum performance. The interface supports power, energy, and temperature measurements of the socket, individual cores, uncore structures, as well as on-chip GPUs. The energy estimates reported by RAPL interface have been validated by Intel to closely follow actual usage. Various tools use RAPL interface to enable power and energy measurements [3].

A modern processor has greater flexibility in varying parameters such as core clock frequency, turbo boost technology, and multi-threaded execution to optimize performance or energy consumption or both [4]. The general assumption is that the highest gains in performance are obtained by executing an application with the maximum possible clock frequency. However, this translates to higher energy consumption and higher operating temperatures. Dynamic frequency scaling, employed in the form of power profiles, is useful when the CPU is idle but is not effective for memory-bounded benchmarks [5]. For data centers that run nonstop the question becomes, what is the optimal core frequency to maximize utilization and minimize energy consumption?

Prior studies suggested that different on-chip structures have separate clock domains dynamically scaled to match application demands [6]. Studies have illustrated the effectiveness of on-chip power meters and explained optimization as a function of performance and energy [7]. Studies conducted to find the optimal frequency range used previous generations of processors and PARSEC benchmarks [8].

This paper aims to provide a clear understanding of how frequency scaling impacts performance, energy efficiency, and a new PE metric that combines both performance and energy efficiency. The measurements are conducted on an Intel Core i7-8700K processor running recently introduced SPEC CPU2017 benchmark suites [9], while varying the number of threads/copies and processor clock frequency. Section II describes the experimental setup and the metrics for evaluation. Section III explains the results obtained and Section IV gives concluding remarks.

## II. EXPERTIMENTAL METHODOLOGY

### A. 2.1 Hardware and Software Setup

The test computer is built around an Intel 8th generation Core i7-8700K. The processor is based on Coffee Lake architecture and is manufactured using Intel's 14nm++ technology node [10]. It consists of six 2-way hyperthreaded physical cores. Each processor core includes separate 8-way set-associative 32 KiB first level caches for instructions (L1I) and data (L1D) and a 4-way 256 KiB unified level 2 cache (L2). The last level cache (LLC or L3) of 12 MiB is shared among all processor cores and is built as a 16-way set-associative structure. The processor's nominal clock frequency is 3.70 GHz. However, turbo boost frequency can reach 4.70 GHz for a single core or 4.3 GHz for all six cores.

The test computer has 32 GiB DDR4 2400MHz RAM memory. The integrated memory controller is configured as dual-channel with a maximum bandwidth of 39 GiB/s. It runs

978-1-7281-0137-8/19/$31.00 ©2019 IEEE

Ubuntu 18.04 LTS with Linux kernel 4.15.0. Hyperthreading is disabled during experiments in this study.

Performance and energy measurements are conducted using a set of lightweight command line tools called likwid (Like I Knew What I'm Doing) [11]. The tools can be roughly grouped into three categories such as system information and control, performance and energy profiling, and micro-benchmarking. Individual tools allow developers to explore memory hierarchy, access performance monitoring counters, control clock frequencies, and control architectural features such as hardware prefetching. Specifically, we use likwid-powermeter, a tool that accesses RAPL counters for measuring power and energy [12], and likwid-setfrequencies, a tool that allows for setting the processor clock frequency.

The SPEC CPU2017 is the latest collection of SPEC's compute-intensive benchmarks with their inputs and run scripts that are designed for measuring and comparing the performance of modern computer systems, stressing their processor, memory subsystem, and compiler. The SPEC CPU2017 contains 43 benchmarks, organized into four suites [13] [14]. The *fp_speed*/*fp_rate* and *int_speed*/*int_rate suites* (Table I and Table II) include benchmarks with predominantly floating-point data and integer data types, respectively, designed to stress speed (*speed* suites) and throughput (*rate* suites) of modern computer systems. The benchmarks written in C, C++, and Fortran programming languages are derived from a wide variety of application domains. The Intel Parallel Studio Cluster XE 2018 is used to compile all the benchmarks with -O3 optimization level [15].

TABLE I. SPEC CPU2017 FLOATING-POINT BENCHMARKS [9]

| SPECrate 2017 Floating Point | SPECspeed 2017 Floating Point | Language | Application Area |
|---|---|---|---|
| 503.bwaves_r | 603.bwaves_s | Fortran | Explosion modeling |
| 507.cactuBSSN_r | 607.cactuBSSN_s | C++, C, Fortran | Physics: relativity |
| 508.namd_r | - | C++ | Molecular dynamics |
| 510.parest_r | - | C++ | Biomedical imaging |
| 511.povray_r | - | C++, C | Ray tracing |
| 519.lbm_r | 619.lbm_s | C | Fluid dynamics |
| 521.wrf_r | 621.wrf_s | Fortran, C | Weather forecasting |
| 526.blender_r | - | C++, C | 3D rendering and animation |
| 527.cam4_r | 627.cam4_s | Fortran, C | Atmosphere modeling |
| - | 628.pop2_s | Fortran, C | Wide-scale ocean modeling |
| 538.imagick_r | 638.imagick_s | C | Image manipulation |
| 544.nab_r | 644.nab_s | C | Molecular dynamics |
| 549.fotonik3d_r | 649.fotonik3d_s | Fortran | Computational Electromagnetics |
| 554.roms_r | 654.roms_s | Fortran | Regional ocean modeling |

A single copy of a *speed* benchmark (name ending with a suffix "_s"), *SBi*, is run on a test machine using the reference input set; the *SPECspeed (SBi)* metric reported by the running script is calculated as the ratio of the benchmark execution times on the reference machine and the test machine as shown in Eq.1.

$$SPECspeed(SBi, N_T) = T(Ref, 1)/T(Test, N_T) \quad (1)$$

A composite single number is also reported for an entire suite; it is calculated as the geometric mean of the individual *SPECspeed* ratios of all benchmarks in that suite. When running speed benchmarks, a performance analyst has an option to specify the number of OpenMP threads, $N_T$, as many benchmarks support multi-threaded execution. Multiple copies ($N_C$) of a rate benchmark (name ending with a suffix "_r"), *RBi*, are typically run on a test machine, and the *SPECrate* (*RBi, $N_C$*) metric is defined as the ratio of the execution times of a single-copy on the reference machine and $N_C$-copy on the test machine, multiplied by the number of copies as shown in Eq. 2

$$SPECrate(RBi, N_C) = N * T(Ref, 1)/T(Test, N_C) \quad (2)$$

TABLE II. CPU2017 INTEGER BENCHMARKS [9]

| SPECrate 2017 Integer | SPECspeed 2017 Integer | Language | Application Area |
|---|---|---|---|
| 500.perlbench_r | 600.perlbench_s | C | Perl interpreter |
| 502.gcc_r | 602.gcc_s | C | GNU C compiler |
| 505.mcf_r | 605.mcf_s | C | Route planning |
| 520.omnetpp_r | 620.omnetpp_s | C++ | Discrete Event simulation: computer network |
| 523.xalancbmk_r | 623.xalancbmk_s | C++ | XML to HTML conversion via XSLT |
| 525.x264_r | 625.x264_s | C | Video compression |
| 531.deepsjeng_r | 631.deepsjeng_s | C++ | AI: alpha-beta tree search (Chess) |
| 541.leela_r | 641.leela_s | C++ | AI: Monte Carlo tree search (Go) |
| 548.exchange2_r | 648.exchange2_s | Fortran | AI: recursive solution generator (Sudoku) |
| 557.xz_r | 657.xz_s | C | General data compression |

### B. Experiments and Metrics for Evaluation

In this study we first characterize each CPU2017 benchmark by measuring its execution time, *T(Bi)*, and energy consumed, *E(Bi)*, when the processor clock is set to the nominal frequency of 3.7 GHz. *Speed* benchmarks are run with $N_T$=1 and $N_T$=6 threads, whereas *rate* benchmarks are run with $N_C$=1 and $N_C$=6 copies. To capture scalability of *speed* benchmarks when running with $N_T$ threads, we define speedup, *S(SBi,$N_T$)*, calculated as shown in Eq. (3), where *T(SBi,1)* and *T(SBi,$N_T$)* are the execution times for a *speed* benchmark *SBi* when run with a single and $N_T$ threads, respectively. The speedup metric for an individual SPEC *rate* benchmark, *RBi*, when run with NC copies, *S(RBi,$N_C$)* is calculated as shown in Eq. (4), where *T(RBi,1)* is the execution time when a single copy of the benchmark is run, and *T(RBi,$N_C$)* is the execution time when $N_C$ copies of the benchmark are run on the test machine. Similarly, we calculate energy efficiency improvement *EE.I*, of each benchmark as shown in Eq. (5) and Eq. (6).

$$S(SBi, N_T) = T(SBi, 1)/T(SBi, N_T) \quad (3)$$

$$S(RBi, N_C) = (N_C \cdot T(RBi, 1))/T(RBi, N_C) \quad (4)$$

$$EE.I(SBi, N_T) = E(SBi, 1)/E(SBi, N_T) \quad (5)$$

$$EE.I(RBi, N_C) = (N_C \cdot E(RBi, 1))/E(RBi, N_C) \quad (6)$$

A combined metric called *PE* is introduced to capture both performance and energy efficiency. The *PE* metric for an individual SPEC *speed* benchmark, *SBi*, running with $N_T$ threads, *PE(SBi, $N_T$)* is defined as shown in Eq. (7), where *E(SBi, $N_T$)* is the processor energy in Joules needed to complete

execution of the benchmark *SBi* when running with $N_T$ threads. To evaluate the performance and energy efficiency of a benchmark run with $N_T$ threads relative to the run with a single thread, an improvement metric defined as shown in Eq. (8) is used. A *PE.I* greater than one means that the benchmark execution with $N_T$ threads improve *PE* metric relative to the corresponding single-threaded execution. The *PE* metric for an individual SPEC *rate* benchmark, *RBi*, running with $N_C$ copies, *PE (RBi,* $N_C$) is defined as shown in Eq. (9). To evaluate *PE* of an $N_C$ -copy benchmark run with respect to a single-copy run, an improvement metric shown in Eq. (10) is used.

$$PE(SBi, N_T) = \quad 1/(T(SBi, N_T) \cdot E(SBi, N_T)) \tag{7}$$

$$PE.I(SBi, N_T) = PE(SBi, N_T)/PE(SBi, 1) \tag{8}$$

$$PE(RBi, N_C) = \quad 1/(T(RBi, N_C) \cdot E(RBi, N_C)) \tag{9}$$

$$PE.I(RBi, N_C) = ({N_C}^2 \cdot PE(RBi, N_C))/PE(RBi, 1) \tag{10}$$

Once the benchmarks are characterized, a series of experiments is conducted to determine the impact of frequency scaling to benchmarks' execution times, energy consumed, and to the *PE* metric. The core clock frequencies are set to the following values: 0.8 GHz, 1.7 GHz, 2.7 GHz, 3.7 GHz, 4.0 GHz, and 4.3 GHz. To facilitate analysis, the metrics of interest, *T*, *E*, and *PE*, are normalized to the corresponding metrics obtained when running at nominal clock frequency of 3.7 GHz. Note that we have monitored core and package temperatures throughout the benchmark execution and found that the highest package temperature was ~57° C while running at 4.30GHz. Hence, we have not experienced thermal throttling.

## III. Results

### A. Performance and Energy Evaluation

Table III shows the execution time (*T*) and energy consumed (*E*), as well as performance speedup (*S*), energy efficiency improvement (*EE.I*), and the *PE.I* metric for $N_T/N_C$=6 when the processor is run at the nominal core frequency, F=3.70 GHz. The results show that the *speed* benchmarks have longer execution times and thus higher energy consumption. Several classes of benchmarks based on their scalability can be observed. The first group includes benchmarks such as *638.imagick_s* and *644.nab_s* that scale extremely well for both performance and energy – *S[6]* is over 5 and *PE.I* is over 11. The second group includes benchmarks such as *627.cam4_s*, *607.cactuBSSN_s*, and *657.xz_s* that scale well – *S[6]* is over 3 and *PE.I* is over 4. The third group consists of benchmarks, such as *621.wrf_s*, *628.pop2_s*, that achieve modest improvements. The last group includes benchmarks that do not scale at all, e.g., *603.bwaves_s*, *619.lbm_s*, *649.fotonik3d_s*, and *654.roms_s*. These benchmarks are memory-bound, and they do not scale when the required memory bandwidth goes beyond the maximum available bandwidth, causing significant losses in energy. Please note that only one benchmark in *int_speed* is parallelizable.

For *rate* benchmarks, *S* and *PE.I* capture throughput scalability when multiple copies of the same benchmark are run concurrently. We can also identify several classes of *fp_rate* benchmarks. The first group includes benchmarks that scale very well with *S[6]*>5 and *PE.I[6]*>11, e.g. *508.namd_r*, *511.povray_r*, *526.blender_r*, *538.imagick_r*, and *544.nab_r*. The second group of benchmarks, e.g., *507.cactuBSSN_r* and *527.cam4_r*, scale well with *S[6]*>4 and *PE.I[6]*>7. The remaining group of benchmarks show limited scalability. The *int_rate* benchmarks scale relatively well. These benchmarks are not memory intensive and do not cause contention on the shared resources. The first group includes the benchmarks that scale very well with *S[6]*>5 and *PE.I[6]*>11, e.g., *500.perlbench_r*, *525.x264_r*, *531.deepsjeng_r*, *541.leela_r*, and *548.exchange2_r*. The second group includes benchmarks with a moderate scalability *S[6]*<5 and *PE.I[6]*<10, e.g., *502.gcc_r*, *505.mcf_r*, *520.omnetpp_r*, *523.xalancbmk_r*, and *557.xz_r*.

TABLE III TIME, ENERGY, PERFORMANCE SPEEDUP, ENERGY EFFICIENCY, AND *PE.I* FOR SPEC CPU2017 BENCHMARKS

| *Benchmarks* | {NT\|NC}=1 | | {NT\|NC}=6 | | S [6/1] | EE.I [6/1] | PE.I [6/1] |
|---|---|---|---|---|---|---|---|
| | T[1] [s] | E[1] [J] | T[6] [s] | E[6] [J] | | | |
| ***fp_speed@3.70GHz*** | | | | | | | |
| **603.bwaves_s** | 1,357 | 23,662 | 845 | 27,889 | 1.61 | 0.85 | 1.36 |
| **607.cactuBSSN_s** | 1,640 | 25,926 | 381 | 14,948 | 4.30 | 1.73 | 7.46 |
| **619.lbm_s** | 833 | 14,828 | 960 | 29,216 | 0.87 | 0.51 | 0.44 |
| **621.wrf_s** | 1,106 | 19,839 | 395 | 15,786 | 2.80 | 1.26 | 3.52 |
| **627.cam4_s** | 1,728 | 28,372 | 444 | 17,628 | 3.90 | 1.61 | 6.27 |
| **628.pop2_s** | 1,231 | 23,335 | 400 | 17,332 | 3.08 | 1.35 | 4.14 |
| **638.imagick_s** | 5,449 | 90,292 | 932 | 44,147 | 5.85 | 2.05 | 11.96 |
| **644.nab_s** | 2,159 | 33,293 | 386 | 15,578 | 5.59 | 2.14 | 11.94 |
| **649.fotonik3d_s** | 693 | 12,810 | 638 | 21,273 | 1.09 | 0.60 | 0.65 |
| **654.roms_s** | 1,706 | 31,291 | 1,118 | 41,466 | 1.53 | 0.75 | 1.15 |
| ***int_speed@3.70GHz*** | | | | | | | |
| **600.perlbench_s** | 305 | 5,097 | 305 | 5,097 | 1.00 | 1.00 | 1.00 |
| **602.gcc_s** | 424 | 6,768 | 424 | 6,768 | 1.00 | 1.00 | 1.00 |
| **605.mcf_s** | 401 | 6,380 | 401 | 6,380 | 1.00 | 1.00 | 1.00 |
| **620.omnetpp_s** | 377 | 5,762 | 377 | 5,762 | 1.00 | 1.00 | 1.00 |
| **623.xalancbmk_s** | 277 | 4,174 | 277 | 4,174 | 1.00 | 1.00 | 1.00 |
| **625.x264_s** | 149 | 2,464 | 149 | 2,464 | 1.00 | 1.00 | 1.00 |
| **631.deepsjeng_s** | 292 | 4,722 | 292 | 4,722 | 1.00 | 1.00 | 1.00 |
| **641.leela_s** | 421 | 6,718 | 421 | 6,718 | 1.00 | 1.00 | 1.00 |
| **648.exchange2_s** | 247 | 4,043 | 247 | 4,043 | 1.00 | 1.00 | 1.00 |
| **657.xz_s** | 2,198 | 32,685 | 659 | 18,872 | 3.33 | 1.73 | 5.77 |
| ***fp_rate@3.70GHz*** | | | | | | | |
| **503.bwaves_r** | 197 | 3,655 | 916 | 30,679 | 1.29 | 0.12 | 0.92 |
| **507.cactuBSSN_r** | 175 | 2,841 | 237 | 9,965 | 4.43 | 0.29 | 7.57 |
| **508.namd_r** | 196 | 3,287 | 197 | 9,621 | 5.98 | 0.34 | 12.27 |
| **510.parest_r** | 304 | 5,480 | 784 | 29,792 | 2.32 | 0.18 | 2.56 |
| **511.povray_r** | 306 | 5,460 | 314 | 16,465 | 5.85 | 0.33 | 11.64 |
| **519.lbm_r** | 86 | 1,853 | 425 | 16,381 | 1.22 | 0.11 | 0.83 |
| **521.wrf_r** | 193 | 3,477 | 417 | 16,816 | 2.79 | 0.21 | 3.46 |
| **526.blender_r** | 248 | 4,034 | 272 | 12,012 | 5.47 | 0.34 | 11.03 |
| **527.cam4_r** | 194 | 3,444 | 269 | 12,332 | 4.33 | 0.28 | 7.25 |
| **538.imagick_r** | 262 | 4,331 | 263 | 12,169 | 6.00 | 0.36 | 12.81 |
| **544.nab_r** | 226 | 3,500 | 227 | 9,509 | 5.97 | 0.37 | 13.19 |
| **549.fotonik3d_r** | 313 | 5,578 | 1,248 | 41,618 | 1.50 | 0.13 | 1.21 |
| **554.roms_r** | 181 | 3,394 | 701 | 26,276 | 1.55 | 0.13 | 1.20 |
| ***int_rate@3.70GHz*** | | | | | | | |
| **500.perlbench_r** | 305 | 5,113 | 337 | 15,590 | 5.43 | 0.33 | 10.69 |
| **502.gcc_r** | 218 | 3,472 | 323 | 12,754 | 4.04 | 0.27 | 6.61 |
| **505.mcf_r** | 217 | 3,413 | 276 | 11,100 | 4.73 | 0.31 | 8.72 |
| **520.omnetpp_r** | 375 | 5,703 | 601 | 21,936 | 3.74 | 0.26 | 5.83 |
| **523.xalancbmk_r** | 277 | 4,191 | 353 | 12,807 | 4.71 | 0.33 | 9.26 |
| **525.x264_r** | 139 | 2,328 | 150 | 7,038 | 5.55 | 0.33 | 11.02 |
| **531.deepsjeng_r** | 235 | 3,798 | 248 | 11,136 | 5.69 | 0.34 | 11.64 |
| **541.leela_r** | 422 | 6,806 | 423 | 18,416 | 5.99 | 0.37 | 13.28 |
| **548.exchange2_r** | 252 | 4,134 | 268 | 11,889 | 5.63 | 0.35 | 11.74 |
| **557.xz_r** | 343 | 5,172 | 414 | 15,777 | 4.97 | 0.33 | 9.78 |

### B. Impact of Frequency Scaling on Performance

Figure I and Figure II illustrate how benchmarks' execution times vary with processor clock frequency when $N_T/N_C$=1 and $N_T/N_C$ =6, respectively. For each benchmark, *Bi*, the ratio of the benchmark's execution time when running at the nominal frequency of 3.7 GHz and the benchmark's execution time when running at frequency *F* is calculated as follows: *T(Bi,3.7GHz)/T(Bi,F)*. This metric is equivalent to the normalized performance for the benchmark *Bi* when running at frequency, *F*, *P(Bi,F)/P(Bi,3.7GHz)*. Straight horizontal lines denote the ratios of processor clock frequencies F/3.7GHz.

For single-threaded/single-copy executions (Figure I), the results indicate that increasing processor clock frequency above the nominal frequency improves performance, but these improvements are rather limited and fall below the ratios of frequencies. For example, when running at F=4.3 GHz, ideally performance gains should match ~4.3/3.7=1.16 or 16%. However, several memory-bounded benchmarks, e.g., *619.lbm_s*, *649.fotonik3d_s*, *503.bwaves_s*, *519.lbm_s*, *549.fotonik3d_r* and *520omnetpp_r*, see no performance gains at all (the green line showing normalized performance when running at 4.3 GHz is below the horizontal pale green line). The geometric means of the performance ratios are ~10% for *fp_speed*, ~14% for *int_speed*, ~12% for *fp_rate*, and ~14% for *int_rate*. Similar behavior can be observed when running at F=4.0 GHz.

These results indicate that increasing processor clock frequency above the nominal frequency is beneficial for benchmarks that are not memory bounded. On the other side, by lowering the processor clock frequency below the nominal frequency, the performance is expectedly reduced. However, the performance losses of the memory-bounded benchmarks are lower than expected. Thus, when running at 2.7 GHz, the expected performance should be ~0.73 (2.7/3.7) of the performance observed at the nominal frequency, or performance losses should be around ~27%. The actual losses for memory-bounded benchmarks are lower than expected, so the geometric means of losses are ~23% for *fp_speed* and ~25% for other suites (the red line showing the normalized performance for all benchmarks when running at 2.7 GHz remains above the pale red horizontal line). Similar findings can be observed when running at even lower clock frequencies of 1.7 GHz and 0.8 GHz.

For multi-threaded and multi-copy benchmark runs ($N_T/N_C$ =6), the results indicate that increases in the clock frequency above the nominal frequency are even less beneficial than in case of single-threaded/single-copy executions, yielding minimal performance gains (see Figure II). For example, the performance gains for *fp_speed* are ~5% (16% is theoretically possible) when running at 4.3 GHz and ~2.5% when running at 4.0 GHz. The performance gains are somewhat better for the *rate* benchmarks reaching ~6.8% for *fp_rate* and ~11.6% for *int_rate* when running at 4.3 GHz. On the other side, by lowering the clock frequency below the nominal frequency, performance losses are significantly lower than expected (e.g., all ratio lines for 2.7 GHz, 1.7 GHz, and 0.8 GHz remain above their respective pale horizontal counterparts).

### C. Impact of Frequency Scaling on Energy Efficiency

Figure III and Figure IV illustrate how the total package energy consumed during benchmark runs varies as a function of the processor clock frequency when $N_T/N_C$=1 and $N_T/N_C$=6, respectively. For each benchmark, *Bi*, the ratio of the benchmark energy when running at the nominal frequency and the energy when running at frequency *F* is calculated as follows: *E(Bi,3.7GHz)/E(Bi,F)*.

The results show that by increasing the processor clock frequency above the nominal, the total energy increases proportionally and thus do not improve energy efficiency even for benchmarks that see significant performance gains. This observation is true for 1T/1C runs as well as for 6T/6C runs. For 1T/1C benchmark runs, using low frequencies of 1.7 GHz and 0.8 GHz is also not beneficial as the total energy increases due to significant increases in the execution times (Figure III). However, when running at 2.7 GHz clock frequency, the total energy decreases compared to the one observed at the nominal frequency. Thus, if the only criterion for optimization is energy-efficiency and performance is not a factor, running at 2.7 GHz is a good choice for clock frequency for 1T/1C benchmark runs (please note that the red line remains above 1 for all benchmarks).

Improvements in the energy efficiency are largest in memory-bounded benchmarks. The total energy gains relative to the nominal frequency are ~8% for the *fp* benchmarks and ~4% for the int benchmarks. Regarding 6T/6C benchmark runs, the results indicate that a lower clock frequency of 1.7 GHz will yield even better energy savings (Figure IV). The improvements in energy efficiency reach 1.57 times relative to the energy efficiency at nominal frequency for *fp_speed*, 1.52 for *fp_rate*, and 1.31 for *int_rate*. Please note that we do not report improvements for *int_speed* as only one benchmark is truly parallel. Somewhat surprisingly even benchmark runs at 0.8 GHz often showed improvements in energy efficiency relative to the energy efficiency of runs with the nominal frequency.

### D. Impact of Frequency Scaling on PE

Figure V and Figure VI show the impact of changes of the processor clock frequency on the combined *PE* metric, when $N_T/N_C$ =1 and $N_T/N_C$ =6, respectively. For each benchmark the ratio of the *PE* when running at the nominal frequency and the *PE* when running at frequency *F* is calculated. The results show that all things considered running at the nominal frequency of 3.7 GHz strikes an optimal balance between performance and energy efficiency for 1T/1C benchmark runs (Figure V).

Some-what surprisingly, the turbo boost modes do not appear to offer clear advantages even in benchmarks that are not bounded by memory. The results in Figure VI indicate that 6T/6C runs at the nominal clock frequency no longer provide an optimal balance between performance and energy for all benchmark suites. Thus, the ratio of the *PE* metric for the nominal frequency and the *PE* for F=2.7 GHz is 1.18 for *fp_speed* benchmarks and 1.15 for *fp_rate* benchmarks. The *PE* gains for integer benchmarks are below 1% and can be considered negligible.

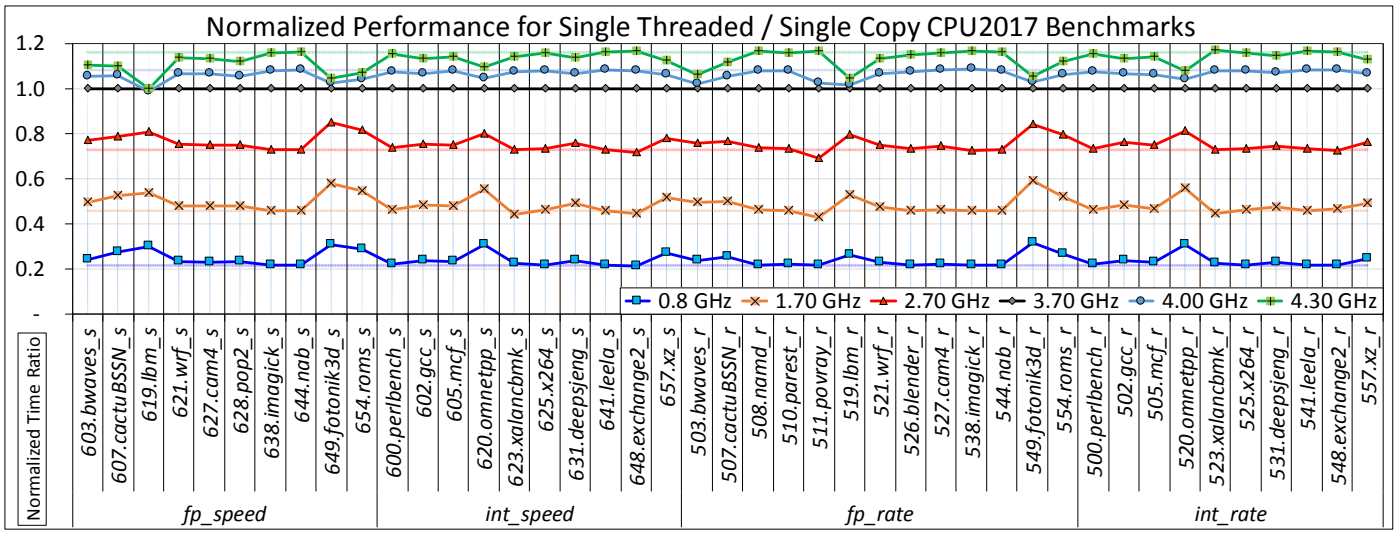


FIGURE I NORMALIZED PERFORMANCE AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\} = 1$

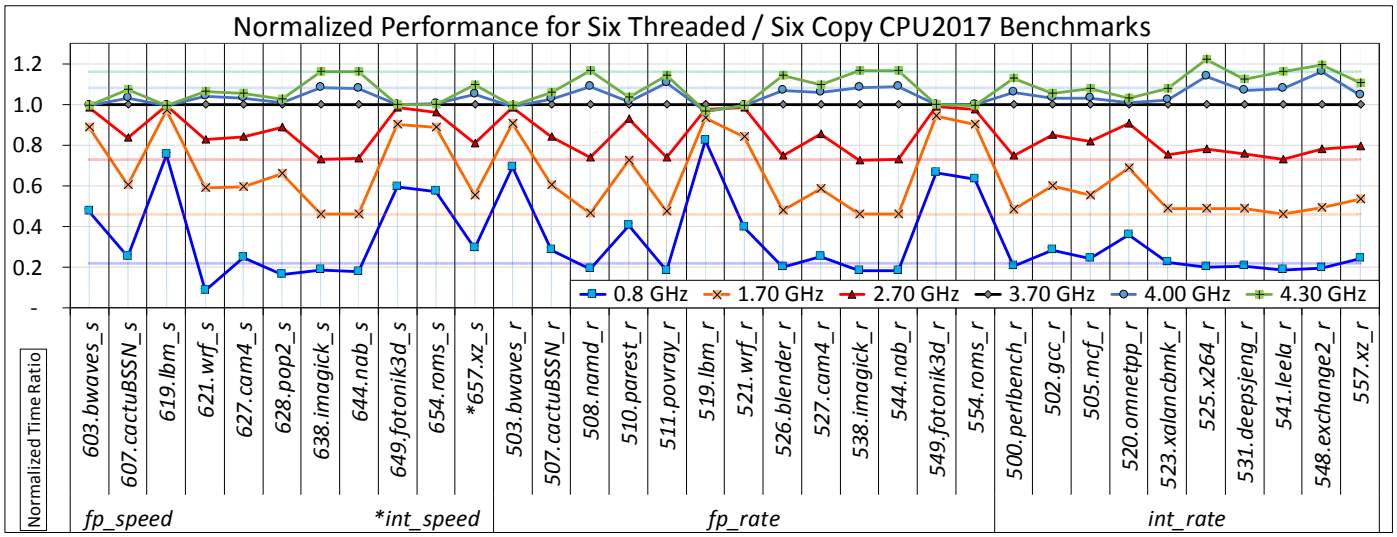


FIGURE II NORMALIZED PERFORMANCE AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\} = 6$

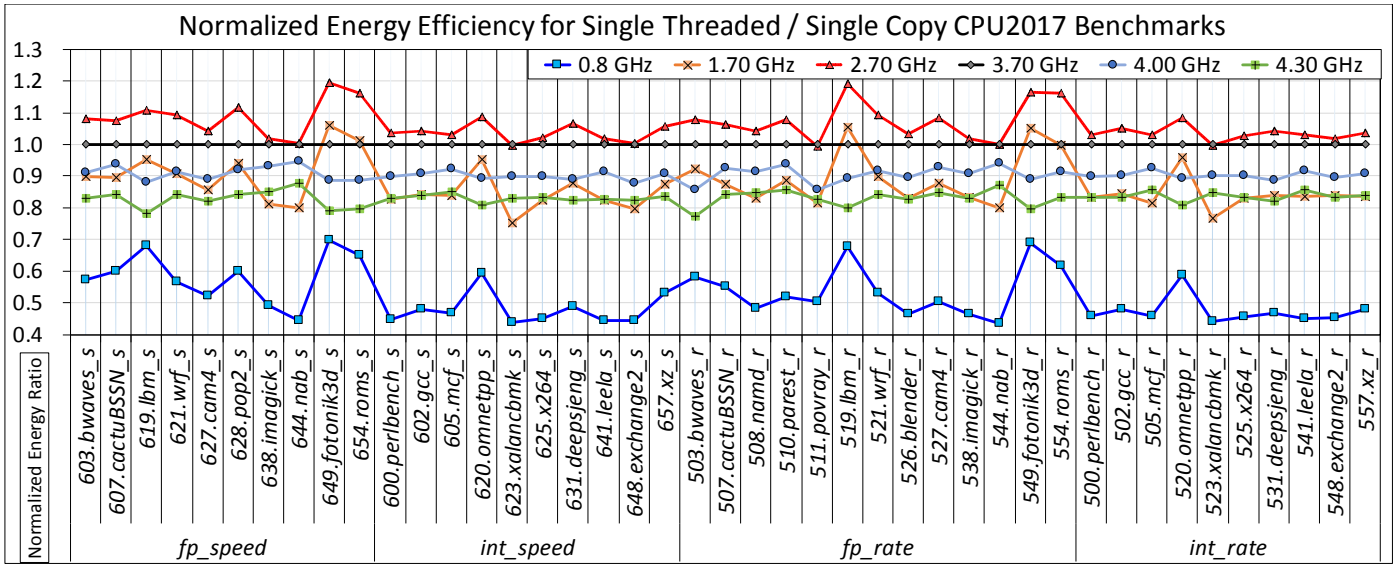


FIGURE III NORMALIZED ENERGY EFFICIENCY AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\} = 1$

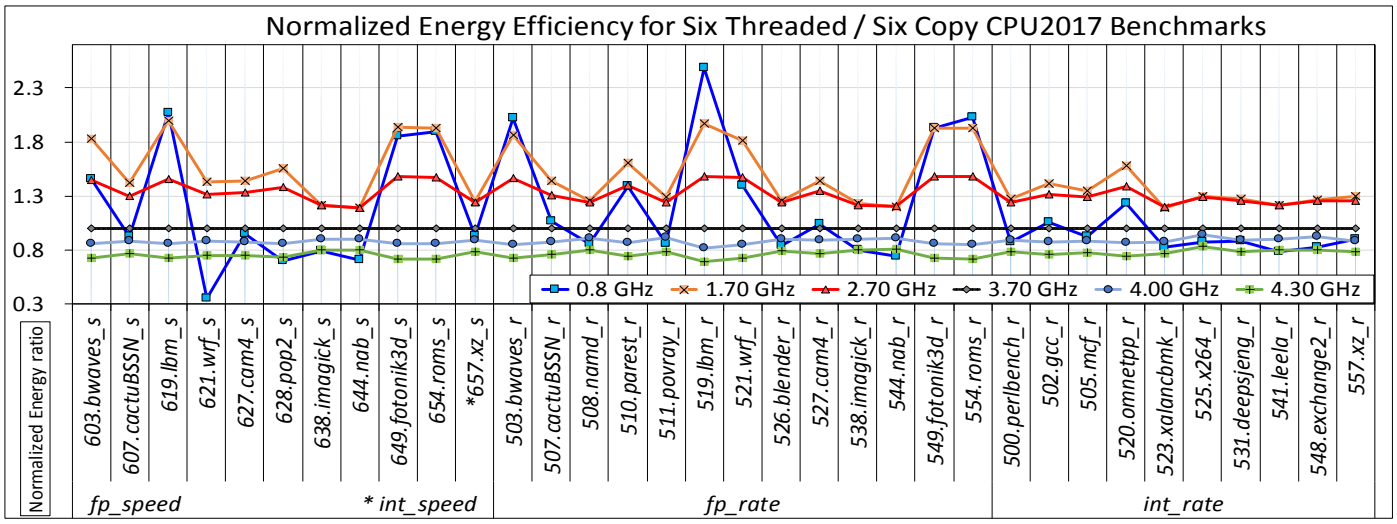


FIGURE IV NORMALIZED ENERGY EFFICIENCY AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\}$ =6

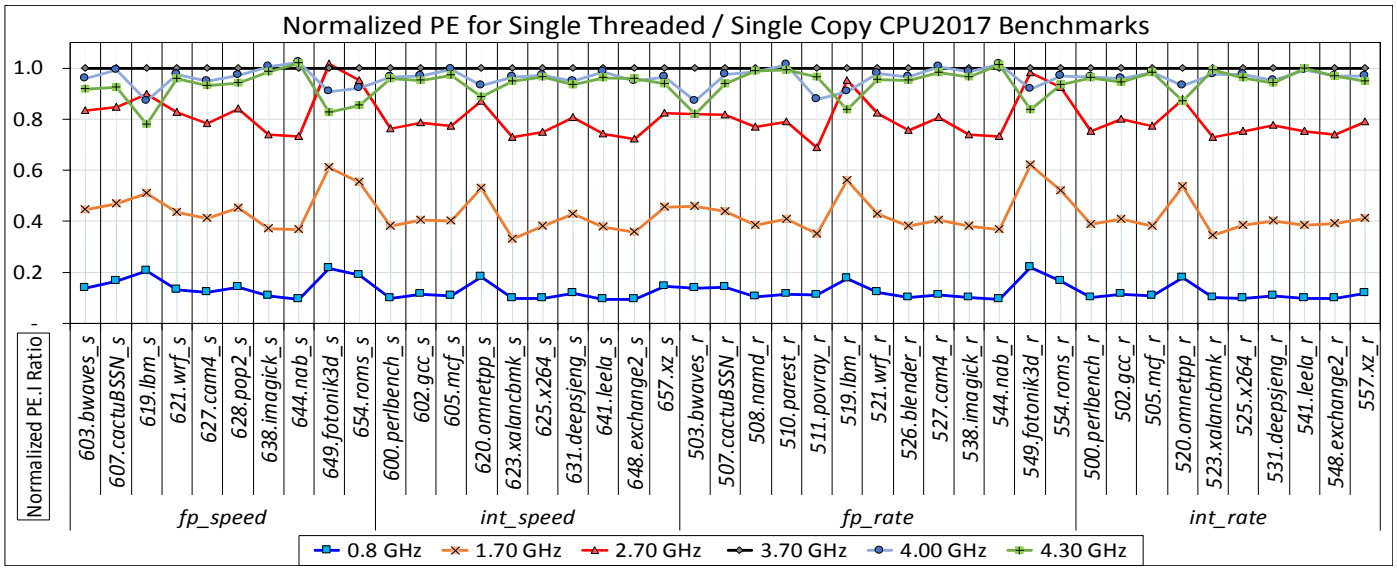


FIGURE V NORMALIZED PE AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\}$ =1

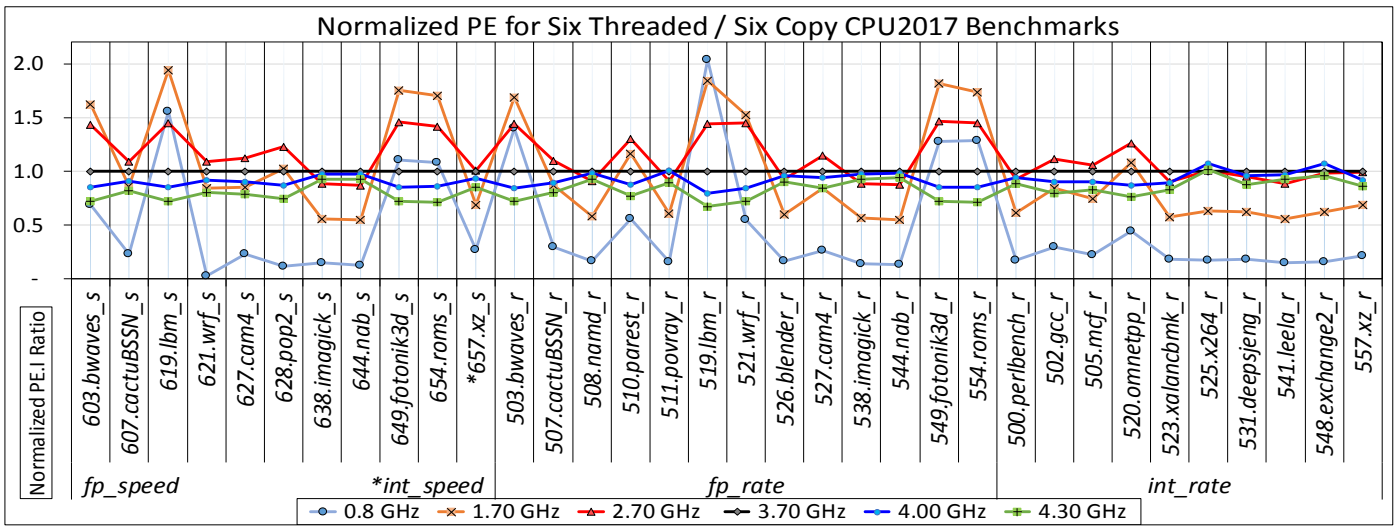


FIGURE VI NORMALIZED PE AS A FUNCTION OF CLOCK FREQUENCY FOR CPU2017 WITH $\{N_T \mid N_C\}$ =1

## IV. Conclusion

The paper shows the results of measurement-based studies focusing on the impact of frequency scaling on performance, energy, and the *PE* metric that captures both performance and energy. The main findings of this work are as follows. (a) Increasing the number of threads/copies to match the number of cores generally improves performance of SPEC CPU2017 benchmarks. (b) For single threaded/copy runs, the best performance is achieved when the clock frequency is at maximum, the best energy efficiency is achieved for an intermediate 2.70 GHz processor clock, and the best *PE* is achieved for the nominal frequency of 3.7 GHz. (c) For multi-threaded/copy runs, the best performance is achieved when the clock frequency is at its maximum, whereas the best energy-efficiency is achieved when the clock frequency is at an intermediate 1.70 GHz, and the best *PE* is obtained when the clock frequency is at 2.70 GHz.

## References


[1] "Here's How Much Energy All US Data Centers Consume," *Data Center Knowledge*, 27-Jun-2016. [Online]. Available: https://tinyurl.com/y96cy9rb. [Accessed: 03-Oct-2018].

[2] "Intel® 64 and IA-32 Architecture's Software Developer's Manual: Vol. 3B," *Intel*. [Online]. Available: https://tinyurl.com/y9rcq29c. [Accessed: 19-Mar-2018].

[3] H. Zhang and H. Hoffmann, "A Quantitative Evaluation of the RAPL Power Control System," *Feedback Computing 2015*, p. 6, 2015.

[4] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An Energy Efficiency Feature Survey of the Intel Haswell Processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 896–904.

[5] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," *2010 HotPower*, Oct. 2010.

[6] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings Eighth International Symposium on High Performance Computer Architecture*, 2002, pp. 29–40.

[7] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking Back on the Language and Hardware Revolutions: Measured Power, Performance, and Scaling ∗," *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, pp. 319–332, Mar. 2011.

[8] A. Dzhagaryan and A. Milenković, "Impact of thread and frequency scaling on performance and energy in modern multicores: a measurement-based study," 2014, pp. 1–6.

[9] "SPEC CPU® 2017." [Online]. Available: https://www.spec.org/cpu2017/. [Accessed: 19-Mar-2018].

[10] "Intel® Core™ i7-8700K Processor Product Specifications," *Intel® ARK (Product Specs)*. [Online]. Available: https://tinyurl.com/ybcw5vc8. [Accessed: 24-Mar-2018].

[11] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments," in *2010 39th International Conference on Parallel Processing Workshops*, 2010, pp. 207–216.

[12] V. M. Weaver *et al.*, "Measuring Energy and Power with PAPI," in *2012 41st International Conference on Parallel Processing Workshops*, 2012, pp. 262–268.

[13] J. Bucek, K.-D. Lange, and J. v. Kistowski, "SPEC CPU2017: Next-Generation Compute Benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering - ICPE '18*, Berlin, Germany, 2018, pp. 41–42.

[14] R. Hebbar Seethur Raviraj, "Spec CPU2017: Performance, Energy and Event Characterization on Modern Processors," M.S.E., The University of Alabama in Huntsville, United States -- Alabama, 2018.

[15] R. Hebbar S R and A. Milenković, "SPEC CPU2017: Performance, Event, and Energy Characterization on the Core i7-8700K," in *10th ACM/SPEC International Conference on Performance Engineering*, Mumbai, India, 2019.