Thermal design power and vectorized instructions behavior

Amina Guermouche, Anne-Cécile Orgerie

Received February 24, 2021; | Revised January 26, 2021; | Accepted February 15, 2021

DOI: xxx/xxxx

CONCURRENCY & COMPUTATION: PRACTICE & EXPERIENCE

Thermal design power and vectorized instructions behavior

Amina Guermouche[1] | Anne-Cécile Orgerie[2]

[1]Télécom SudParis, Institut Polytechnique de Paris, Evry, France
[2]Univ. of Rennes, Inria, CNRS, IRISA, Rennes, France

Correspondence
*Amina Guermouche Email: amina.guermouche@telecom-sudparis.eu

Vectorized instructions were introduced to improve the performance of applications. However, they come at the cost of an increase in the power consumption. As a consequence, processors are designed to limit their frequency when such instructions are used in order to respect the thermal design power limit.

In this paper, we study and compare the impact of thermal design power and SIMD instructions on performance, power and energy consumption of processors and memory. The study is performed on three different architectures providing different characteristics and four applications with different profiles (including one application with different phases, each phase having a different profile).

The study shows that, because of processor frequency, performance and power consumption are strongly related to thermal design power. It also shows that AVX512 has unexpected behavior regarding processor power consumption, while DRAM power consumption is impacted by SIMD instructions because of the generated memory throughput. Finally, this paper tackles the impact of turboboost which shows equivalent to better performance for all the studied cases while not always decreasing energy consumption.

KEYWORDS:
Power consumption; energy efficiency; SIMD instructions; TDP; memory; turboboost.

1 | INTRODUCTION

The race for computing performance has led the computer vendors to introduce many features and new techniques to run computations as fast as possible. Such hardware improvements has allowed the supercomputers in the TOP500 to gain 6 orders of magnitude in terms of performance in the last 25 years (1).

Turboboost, one of these improvements, enables the processor to run at higher frequencies than the base one in order to increase performance. Simple Instruction Multiple Data (SIMD) model is another of these hardware techniques. In the SIMD model, the same operation is executed simultaneously on different elements of a vector or different data points. For instance, several iterations of the same loop for a vector/vector addition can be processed at once. Figure 1 shows the difference between an SIMD processor and a scalar processor.

This performance race comes with side effects in terms of power consumption and heat dissipation. Indeed, the power efficiency of supercomputers in the TOP500 has gained 3 orders of magnitude in the last 15 years – this metric being collected since fewer years than performance (1). It means that the power consumption of supercomputers keeps increasing, resulting in larger and larger heat dissipation at the processor level. Consequently, because of thermal limits, it leads to a growing fraction

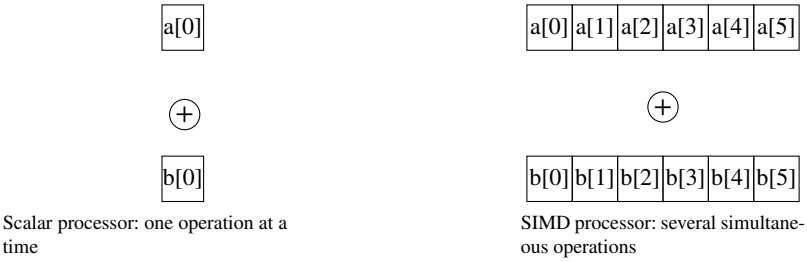

FIGURE 1 Difference between SIMD and scalar operations. Example: addition of two vectors

of dark silicon (2) in modern architectures. To prevent physical damages due to heat, processors are designed to respect a thermal design power (TDP). TDP is the average power that the processor dissipates when operating at the base frequency with all cores active (3). The base frequency is defined as the frequency when turboboost is disabled.

The performance of hardware improvements can be limited by the TDP. Indeed, when the processor detects SIMD instructions, additional voltage is applied to the core. With the additional voltage applied, the processor could run hotter, requiring the operating frequency to be reduced to maintain operations within the TDP limits. This is also the case for turboboost frequencies (4).

The TDP limit enforcement impacts the execution of applications in a non trivial manner. As an example, Table 1 provides the execution time when running HPL, a CPU-intensive HPC benchmark, with and without turboboost on server chifflet (this experiment is described in details later in the paper). It illustrates that, unexpectedly, turboboost does not significantly increase the performance (0.32% difference) on this server for this application, which is however optimized for vectorized instructions.

	with Turboboost	without Turboboost
Execution time (s.)	802.92	805.504

TABLE 1 Execution time of HPL on chifflet with AVX2 while enabling and disabling turboboost (average on 5 runs).

The goal of this paper is to experimentally study such behaviors and provide scenarios where unexpected results may be obtained. The idea consists in identifying the impact of thermal design power on SIMD instructions. In order to do so, we will examine the behavior of different application profiles, when using different SIMD instructions on different hardware platforms. Note that we will only focus on Intel architectures in the remainder of the paper. This study aims at analyzing and comparing the power and energy consumption of each type of instruction and presenting cases of abnormal and unexpected behaviors. Among the unexpected behaviors, this study provides actual examples illustrating that faster runs do not necessarily imply less energy consumption, and that using turboboost on a given architecture can save time and energy for different applications, while disabling turboboost consumes more time but less energy on another hardware architecture for the same applications. Moreover, this study exhibits experiments to deal with turboboost in depth, unlike other studies in the literature that start by deactivating the turboboost because of its impacts on performance and power consumption (5, 6). All the data collected from the experiments of these study and presented hereafter are publicly available: https://gitlab.inria.fr/orgerie/greenavx-data.

The paper is organized as follows. Section 2 describes the experimental testbed. The used applications are detailed in Section 3 and experimental results are analyzed in Section 4. Related work is presented in Section 5. Section 6 concludes this experimental analysis.

2 | ARCHITECTURE AND EXPERIMENTAL TESTBED

This section briefly presents SIMD instructions characteristics. Then it describes the processors used for the experiments and the power measurement methodology. All the obtained raw data are made available: https://gitlab.inria.fr/orgerie/greenavx-data.

2.1 | SIMD instructions

As stated before, SIMD instructions allow the same operation to be executed simultaneously on different elements of a vector or different data points. The number of simultaneous operations depends on the registers' size provided by the processors. Intel implements floating point SIMD instructions since the late 90's with the introduction of Streaming SIMD Extensions (SSE). Registers of 128 bits (16 Bytes) were used to hold 2 double-precision floats or 4 single-precision floats. Advanced Vector Extensions (AVX) appeared in 2010. The registers' size was doubled (256 bits) for floating point operations. However, the 128-bits integer SIMD instructions were not expanded. Finally, since the Haswell processor (2013), AVX2 extensions were introduced. They expand most 128-bits SIMD integer instructions to 256 bits. Moreover, AVX2 extension adds fused multiply-add instructions (FMA) (7). AVX-512 is a 512-bits extension of the 256 bits operations (for floating point and integer operations). They are implemented in the Intel Xeon Phi and Skylake CPUs (2015).

2.2 | Target platform

For the experiments, we used three servers from the Grid'5000 (8) platform (nova, chifflet and yeti). We chose these nodes because they do not provide the same characteristics: nova runs at the same frequency regardless of the SIMD instructions being used. Chifflet and yeti frequencies are impacted by SIMD instructions. Besides, yeti provides AVX512 instruction set. The nodes are described below and their characteristics in terms of TDP and frequencies are summarized in Table 2 :

- nova: The Nova cluster, located in Grid'5000 Lyon site, is equipped with 23 Intel(R) Xeon(R) CPU E5-2620 v4. All experiments were run on nova-11. It is equipped with 2 CPUs, 8 cores per CPU and 64GB of memory. It provides SSE, AVX and AVX2 instruction sets. Note that for this processor, turboboost frequency does not depend on the SIMD instruction being used.
- chifflet: Chifflet is a cluster located in Grid'5000 Lille site and is equipped with 8 Intel Xeon E5-2680 v4. We used chifflet-1 for our experiments. It is equipped with 2 CPU, 14 cores per CPU and 768 GB of memory. It provides SSE, AVX and AVX2 instruction sets. For this processor, the frequency varies according to the SIMD instruction used (as shown in Table 2).
- yeti: Yeti is a cluster located in Grid'5000 Grenoble site. It has 4 Intel(R) Xeon(R) Gold 6130, each equipped with 16 cores. Each NUMA node has 63 GB of memory. Thus, yeti has 64 cores and 252GB of memory in total. It provides SSE, AVX, AVX2 and AVX512 instruction sets. Like chifflet, SIMD instructions have an impact on the turboboost frequency. Moreover, AVX512 has also an impact on the base frequency as shown in Table 2 . Note that we used yeti-2 for all our experiments.

All platforms run under Intel Pstate with performance governor. Table 2 provides the idle and base frequency, and characteristics for each server. It also shows the turboboost frequency depending on the SIMD instruction. Note that the turboboost frequency is not the frequency of the processor during the whole execution, but rather the frequency when SIMD instructions are used. Thus, higher frequencies may be observed during the execution. On nova, the turboboost frequency is independent from the SIMD instructions being used while yeti shows the same base and turboboost frequencies when using AVX512. Moreover, when reaching TDP, processor may run at lower frequencies on chifflet and yeti as we will observe in Section 4. Yeti data were extracted from (9) while nova and chifflet data can be found in (10). One should note that while TDP constitutes a physical hard limit, a processor can briefly goes beyond it due to thermal inertia.

2.3 | Power measurements methodology

In our experiments, we measure the power consumption and the execution time. Execution time is provided by the applications themselves.

In our study, power measurements rely on LIKWID (11) (version 4.3.0[1]). LIKWID is a set of command line tools that are able to read performance counters, pin threads, ... In order to measure the power consumed during the execution of an application, we use `likwid-perfctr` which reads the corresponding hardware counters. We measure both sockets and DRAM power consumption of the target platforms. Note that the words package, socket or processor will be used in the document. `likwid-perfctr` relies on Running Average Power Limit (RAPL) counters. RAPL was introduced by Intel in order to stay

[1] LIKWID commit: d4deec4ca7769a8fbea4a71bcf2cdad9e5b644a2

	nova	chifflet	yeti
number of cores	16	28	64
idle frequency (GHz)	1.2	1.2	1
base frequency (GHz)	2.1	2.4	2.1
TDP (W) per socket	85	120	125
AVX512 Base frequency (GHz)	-	-	1.9
Turboboost SSE frequency (GHz)	2.3	2.9	2.8
Turboboost AVX frequency (GHz)	2.3	2.8	2.5
Turboboost AVX2 frequency (GHz)	2.3	2.8	2.4
Turboboost AVX512 frequency (GHz)	-	-	1.9

TABLE 2 Target platforms characteristics extracted from processors documentation

within the power limit. It uses Dynamic Voltage and Frequency Scaling (DVFS) to guarantee the desired power limit. RAPL interface describes Model Specific Registers (MSRs). These registers provide energy consumption information for components such as the processor and the DRAM. On processors like Intel Sandy Bridge, RAPL was based on a modeling approach. Since the Intel Haswell generation, the processors have fully integrated voltage regulators (FIVR) that provide actual measurements and allow for more accuracy (12). Note that the literature provides many studies on RAPL accuracy for both DRAM (12, 13) and processor (14).

In all our experiments, except for AFF3CT, the measurements are performed every second, with no overhead for all the applications. The mean power consumption is computed over the whole execution time. Regarding AFF3CT, its execution time being too short on yeti, we take the total power and energy consumption provided by LIKWID, rather than a measurement every second.The mean of the power consumption is computed over the whole execution. The results presented in Section 4 represent the average over 5 runs. Regarding measurements errors, all configurations show a small standard deviation. The maximum package power difference observed was on yeti with SVD_Bulge using AVX512 (2W over 122W). The variation in DRAM measurements are very low ($< 1\%$).

3 | APPLICATIONS

This section describes the applications and the configuration parameters that we used.

In order to study AVX impact on power and energy consumption, we target 4 different HPC applications, which have automatic vectorization (by setting a compilation flag or an environment variable). We used applications with different CPU behavior and/or available options: HPL (15) and Plasma svd (16) which use the Math Kernel Library (MKL), AFF3CT (17) and SToRM (18). Note that AFF3CT and SToRM only use integer. The following paragraphs present a short description of the applications and their configurations.

- High Performance Linpack (HPL) (15): HPL is a software package that solves dense linear algebra systems. It is used as a reference benchmark to compute the performance of the supercomputers in the TOP500 (1). All the parameters are presented in Table 3 a. We used HPL version 2.3 compiled with OpenMPI 3.1.4 and with MKL library version l_mkl_2018.3.222. MKL allows choosing the right SIMD instructions by setting the environment variable MKL_ENABLE_INSTRUCTIONS to SSE4_2, AVX, AVX2 or AVX512 (for yeti only). Note that HPL is a CPU-intensive application.

- PLASMA Singular Value Decomposition (SVD) (16): The SVD decomposition computes the singular values of a matrix. It is performed in three steps. Readers can refer to (16) for more details on the algorithm. The first step is referred to as SVD_Band in the remainder of the paper while the second is referred to as SVD_Bulge. In our configuration, we used the SVD version implemented in PLASMA (19). PLASMA is a software package for solving problems in dense linear algebra using multicore processors and Xeon Phi coprocessors. We used PLASMA version 2.8.0 compiled with MKL library version l_mkl_2018.3.222. Thus, setting the desired SIMD instruction is done the same way as HPL. Table 3 b details the parameters values used for the target nodes. We fixed the size such that the first two phases last long enough to have several power measurements. The first phase is CPU-intensive while the second phase is memory-intensive. Note that the third phase is very short (few seconds) for these configurations. For this reason, we will not present an analysis for this phase.

	N	NB	(PxQ)
nova	58912	224	(4x4)
chifflet	100996	224	(4x7)
yeti	91840	224	(8x8)

HPL

	N	NB
nova	20000	200
chifflet	28000	200
yeti	48000	200

SVD

TABLE 3 HPL and SVD setup for nova, chifflet and yeti

- Seed-based Read Mapping tool for SOLiD or Illumina sequencing data (SToRM) (18): SToRM is a read mapping tool based on mapping data between reads and a reference genome. It runs several phases. The first phase loads a sequence from a database. Then the search algorithm is applied before generating the output.We present the results for the search algorithm used in the application. As SToRM uses integers in the SIMD parts, only SSE, AVX2 and AVX512 results will be presented. Note that SToRM is a CPU-intensive application. We will no further detail the software. The user can refer to (18) for more details. Section 7.4 is an appendix detailing how we generated the input reads for SToRM. Note that the SIMD instructions are handled within the code. Setting the desired SIMD instruction set is done using the right compilation flag.
- A Fast Forward Error Correction Toolbox (AFF3CT) (20): AFF3CT[2] is a library used for forward error correction. Forward Error Correction (FEC) is used to control errors during data transmission in order to enable efficient communications over noisy channels. It is done through encoding (by the sender) the data frame and decoding (by the receiver). We will no further detail Forward Error Correction, but we will present how the authors used vectorized instructions in their decoding solution. The decoder takes a set of frames as input. The decoding of the frames is vectorized. In order to do so, the frames are first buffered and then the vectorized algorithm is applied on the frames. Thus, depending on the SIMD instruction being used, the number of loaded frames differs. For instance, when using AVX2 instructions, twice the number of frames are loaded simultaneously compared to using SSE instructions. Just like SToRM, the vectorization is handled within AFF3CT code and a compilation flags allows setting the desired SIMD instruction. Note that this application is CPU-intensive and uses integers which are not supported by AVX instructions. Thus, there will be not results for AVX with AFF3CT.

On all platforms, applications were compiled against gcc 6.3.0 and the machines are running a 4.9.168-1+deb9u2 (2019-05-13) x86_64 GNU/Linux. On all architectures, we used all available cores while disabling hyperthreading.

3.1 | Number of LIKWID measurements

Table 4 details the number of LIKWID measurements of each application under the different configurations. Recall that for AFF3CT, we do not perform a measurement every second but gather the total power consumption at the end of the execution. As a consequence, AFF3CT is not shown in Table 4 . As the table shows, most applications have more than 50 measurements except SVD_BAND which has 37 measurements on yeti. Since the variation is very small as indicated in section 2.3, we believe that the number of measurements is enough to guarantee the accuracy of the results.

3.2 | Applications vectorization rate

To get an idea on how vectorization impacts the target applications, we used the event group FLOPS_DP from LIKWID. It provides the number of scalar and vectorized floating point operations in addition to the vectorization ratio. Table 5 shows the vectorization ratio for HPL, SVD_BAND and SVD_BULGE. All measurements are performed on nova. HPL reaches 99 % of vectorization over all the floating point operations. The same behavior is observed for SVD_BAND and SVD_BULGE. This indicates that, despite being memory intensive, SVD_BULGE manages to fully benefit from vectorization during its floating point operations computations. Note however that the number of issued floating point operations with AVX2 reaches 21 % of the total number of issued operations for SVD_BULGE, while it reaches 88 % for SVD_BAND.

AFF3CT and SToRM performs integer operations. As a consequence, we cannot rely on performance counters since there is no counter which provides integer performance. In order to have an intuition of how many instructions of SToRM and AFF3CT

[2] git tag 1ceddfc874f38317e83592b506b1fe78dffe44b4

	application	SSE	AVX	AVX2	AVX512
nova	HPL	1008	520	284	-
	SVD_Band	181	107	60	-
	SVD_Bulge	53	53	50	-
	SToRM	200	-	119	-
chifflet	HPL	2311	1307	801	-
	SVD_Band	232	144	89	-
	SVD_Bulge	56	57	53	-
	SToRM	85	-	51	-
yeti	HPL	849	481	274	185
	SVD_Band	531	361	237	213
	SVD_Bulge	41	42	37	37
	SToRM	365	-	205	183

TABLE 4 Number of LIKWID measurements for each application on nova, chifflet and yeti (with one measurement per second).

application	SSE	AVX2
HPL	99.89	99.78
SVD_BAND	99.91	99.98
SVD_BULGE	99.11	97.80

TABLE 5 : Vectorization ratio on nova

are vectorized, we used LIKWID to measure the executed micro operations (UOPS group) with AVX2 and SSE. Since AVX2 instructions use larger registers compared to SSE, they should require less operations to be executed at once. Table 6 shows the ratio SSE over AVX2 of the total number of retired UOPS (number of completely executed micro-operations). We use HPL as a comparison basis since it has a large vectorization ratio. On the other hand, an application without vectorization should not see a variation in its number of operations. The ratio should be 1 in this case which we consider as the lower bound of the ratio. Table 6 shows that both AFF3CT and SToRM exhibit a difference in the number of executed UOPS. The ratio is above 1.5 for both applications. This indicates that both applications are impacted by vectorization but less than HPL.

application	SSE / AVX2
HPL	3.31
AFF3CT	1.67
SToRM	1.81

TABLE 6 : Retired UOPS of SSE over AVX2 on nova

4 | EXPERIMENTAL RESULTS

This section presents SIMD instructions behavior when the power consumption reaches TDP and when it does not. To do so, for each application and each platform, we use the available SIMD instructions on each machine. For each configuration, we will explain the observed behavior of SIMD instructions and if TDP is impacting this behavior. This study is on applications socket power consumption (Section 4.1), performance (Section 4.2), energy consumption (Section 4.3) and DRAM power (Section 4.4) and energy (Section 4.5) consumption. The goal here is not to compare the different platforms, but

rather to study the behavior of SIMD instructions under different configurations. All the raw values are available online: https://gitlab.inria.fr/orgerie/greenavx-data.

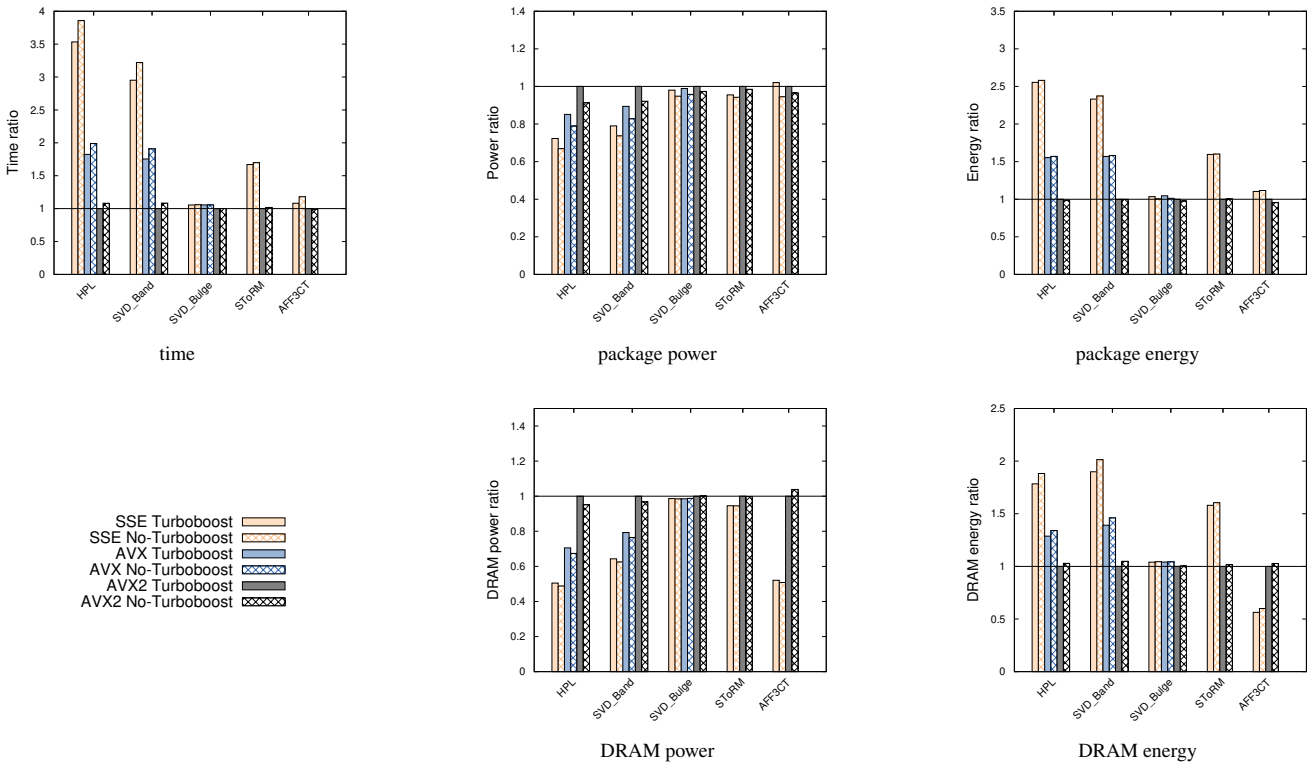

FIGURE 2 Comparison of SIMD instructions with and without turboboost on nova

Figure 2 shows the execution time ratio (Figure 2 a), socket power ratio (Figure 2 b), socket energy consumption ratio (Figure 2 c), DRAM power ratio (Figure 2 e) and DRAM energy consumption ratio (Figure 2 f) of the different applications on nova while using the different SIMD instructions when turboboost is activated and not activated. Figures 3 and 4 show the same ratios for chifflet and yeti. For each application, the ratio is computed over the default SIMD instruction which is AVX2 for nova and chifflet and AVX512 on yeti.

4.1 | Impact on applications socket power consumption

In this section, we present how SIMD instructions impact power consumption. We describe the results for each platform separately since they are not similar.

4.1.1 | nova: SIMD at the same frequency

On nova (Figure 2 b), for all CPU-intensive applications except AFF3CT, the larger the SIMD instruction vector size, the larger the power consumption. This is the typical behavior described in the literature (7). Regarding AFF3CT, SSE has a larger power consumption compared to AVX2. This is because of the frequency. This can already be verified since disabling turboboost shows the normal behavior where AVX2 has a larger power consumption compared to SSE. We also measured the frequency during the execution of AFF3CT. Unlike what is claimed in (10), a small frequency variation can be observed on nova as well. As a matter of fact, when running AFF3CT with AVX2, the average observed frequency is 2.27 GHz while it is 2.3 GHz with SSE. Note that for the other applications, the frequency remains constant as stated in section 2.2.

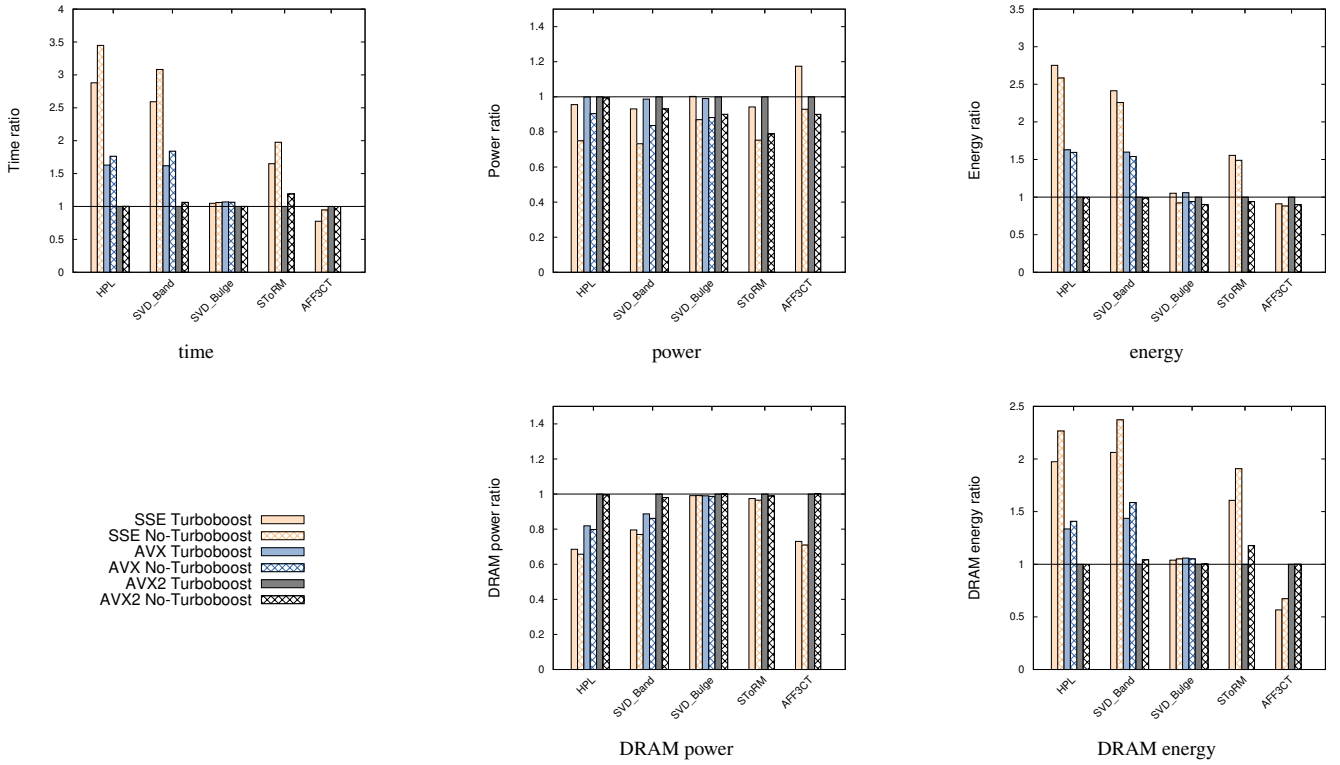

FIGURE 3 Comparison of SIMD instructions with and without turboboost for on chifflet

application	SSE	AVX	AVX2
HPL	2.79	2.63	2.44
SVD_Band	2.89	2.79	2.63
SVD_Bulge	2.75	2.69	2.66
SToRM	2.89	-	2.89
AFF3CT	2.90	-	2.74

TABLE 7 Average observed frequency, in GHz, on chifflet over all cores.

Regarding SVD_Bulge, which is memory-intensive, one can see that SIMD instructions have a very small impact on power consumption (an increase of 1.99% from SSE to AVX2 and 1.08% from AVX to AVX2). This slight impact is explained by the small impact on the performance as shown in Figure 2 a which indicates the use of SIMD instructions in SVD_Bulge.

Finally, disabling turboboost shows the same behavior. As the frequency is lower, the power consumption when disabling turboboost is lower for all applications which is the normal behavior. As expected, for all applications, the power ratios of all instructions over AVX2 when enabling turboboost are the same as the ratios when turboboost is disabled. This indicates that for nova, since all instructions run at the same frequency, the SIMD instruction being used is the major factor impacting power consumption and shows that, as expected, as the registers size increases, the power consumption increases.

4.1.2 | chifflet: SIMD at different frequencies

On chifflet (Figure 3 b), the behavior is similar to nova: the larger the SIMD registers size, the larger the power consumption for all CPU-intensive applications except AFF3CT. However, the difference between the power consumption of the different instructions is less important on chifflet compared to nova for HPL and SVD_Band. As a matter of fact, on nova using AVX2 consumes 38.33% more than using SSE for HPL and 26.58% more for SVD_BAND. On the other hand, on chifflet, using AVX2 only consumes 4.63% more than using SSE for HPL and 7.36% more for SVD_BAND.

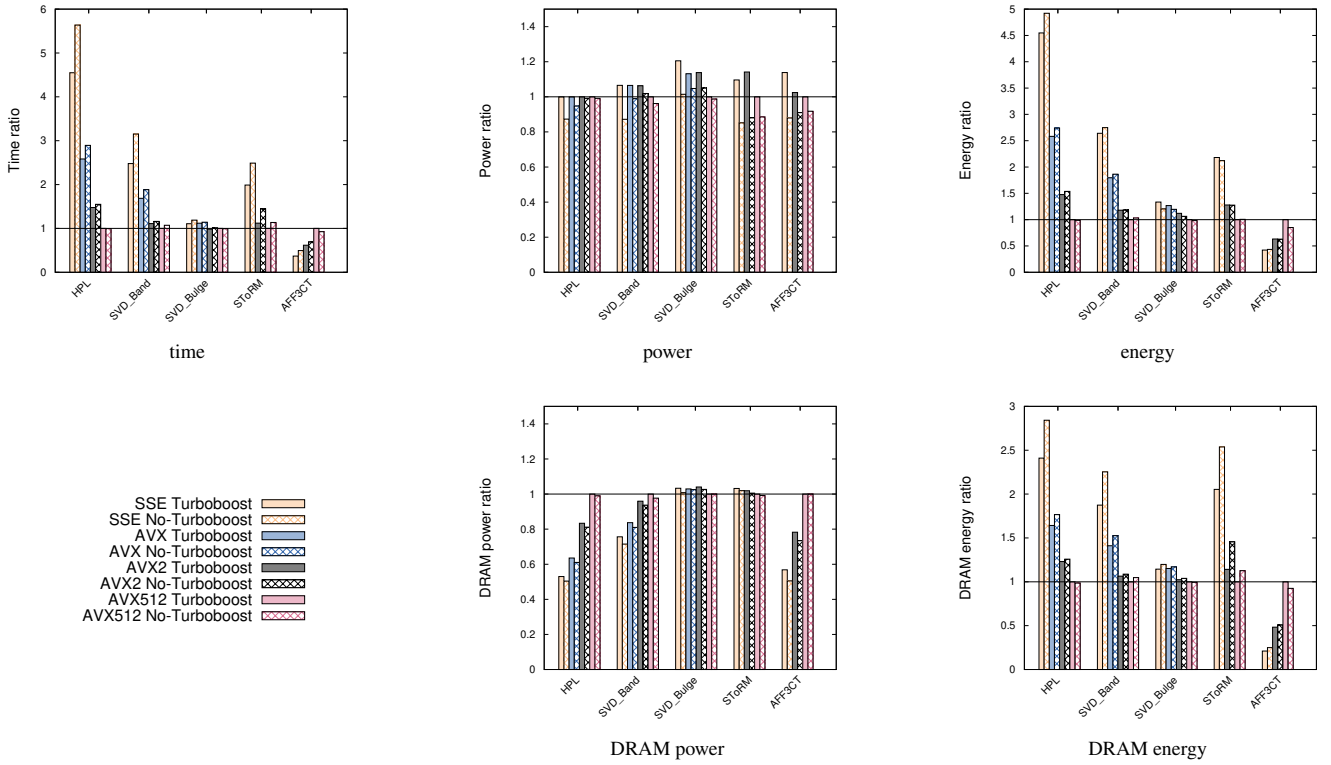

FIGURE 4 Comparison of SIMD instructions with and without turboboost for on yeti

application	SSE	AVX	AVX2	AVX512	AVX512 noTB
HPL	2.67	2.4	2.30	1.95	1.85
SVD_Band	2.74	2.41	2.41	2.07	1.94
SVD_Bulge	2.71	2.43	2.42	1.93	1.91
SToRM	2.80	-	2.79	2.4	2.1
AFF3CT	2.81	-	2.66	2.08	2.02

TABLE 8 Average observed frequency, in GHz, on yeti over all cores.

For these two applications, AVX is almost at the thermal design power while AVX2 reaches it. Because of that, for HPL and SVD_Band, the frequency is reduced for AVX and AVX2. This explains why they have roughly the same power consumption. Tables 7 and 8 show the observed frequencies for all applications on chifflet and yeti respectively. For HPL, AVX2 frequency is equal to the base frequency (2.4GHz) while AVX frequency is 2.6GHz. This explains why disabling turboboost shows the same values for performance and power consumption for AVX2 (as hinted in Table 1). For SVD_Band, the frequency is 2.6GHz for AVX2 and 2.79GHz for AVX. Thus, unlike HPL, AVX2 does not run at the base frequency, which explains the difference in performance and power consumption between AVX and AVX2 for SVD_Band compared to HPL.

Regarding SVD_Bulge, SIMD instructions have no impact on the power consumption. Figure 3 b shows that AVX consumes slightly less power than to AVX2 or SSE, but this difference is 1.26%, which is in the error measurement range. When setting all processors to the same frequency (No-Turboboost plot), one can see a slight difference between the instructions (5.8W for SSE and 3.5W for AVX compared to AVX2). This is due to the use of SIMD instructions in SVD_Bulge (as stated in section 3.2 and will also be shown in section 4.2). However, the variation is small (at most 3.5%).

SToRM shows an interesting behavior. According to Table 7 , AVX2 reaches the same frequency as SSE. Thus, the ratio SSE/AVX2 is the same regardless of turboboost. Moreover, just like AFF3CT, it uses only integers, which means that for all instructions, only half of the registers size is used. This impacts power consumption. As a matter of fact, SToRM consumes 101.82 W per socket while HPL consumes 120 W for AVX2. Note that we do not compare the applications since they may not

have the same computation intensity, we only state that using half of the registers size most likely leads to less power consumption and thus higher frequencies can be used.

Finally, AFF3CT has a different behavior on chifflet compared to nova: Unlike the other applications, the larger the SIMD registers size, the lower the power consumption even when turboboost is disabled (while on nova, this behavior is observed only when turboboost is enabled). For instance, using SSE consumes 14.86% more power than using AVX2. This is due to the fact that SSE outperforms AVX2 for this application. Disabling turboboost shows a similar behavior but with a smaller difference as both instructions run at the same frequency. As a matter of fact, SSE power consumption is 3.15% higher than AVX2.

4.1.3 | yeti: AVX512

Applications on yeti exhibit a completely different behavior compared to the other platforms: larger SIMD registers does not mean more power consumption. This means that AVX512 shows the same or less power consumption compared to other instructions. This is different from the behavior observed on nova and chifflet.

On Figure 4 b, for all applications, the power consumed when using AVX512 (thus SIMD instruction with the largest registers) is lower or equal (equal for HPL) compared to using other registers. The reason why SSE, AVX and AVX2 consume more power than AVX512 is that they reach higher frequencies as shown in Table 8 . This can be already be observed for SToRM when disabling turboboost. We will start by describing the behavior for this application before moving to HPL, SVD and AFF3CT.

SToRM has a specific behavior on yeti: AVX2 and AVX512 do not run at the same frequency as specified in Table 2 . As a matter of fact, AVX2 frequency is the same as SSE whereas AVX512 frequency is 2.4GHz when turboboost is enabled. AVX2 reaches the thermal design power and consumes more power than SSE (by 4.1%). Besides, SSE consumes 9.64% more than AVX512 because of AVX512 frequency. This can be verified when observing the results when turboboost is disabled. In this case, unlike the other applications, AVX512 frequency is the same as SSE and AVX2. As a consequence, since all instructions run at the same frequency, AVX512 consumes 4% more power than SSE. AVX2 and AVX512 have roughly the same power consumption when disabling turboboost.

AFF3CT shows a behavior where SSE and AVX2 consume more power compared to AVX512. This is due to both performance (since AVX2 performance are better than AVX512, its power consumption is larger) and frequency since disabling turboboost shows that SSE consumes slightly less than AVX2 which consumes slightly less than AVX512.

For HPL, all SIMD instructions reach the TDP which is why they have the same power consumption. Even when turboboost is disabled, AVX2 is almost at TDP.

Regarding SVD_Band, SSE, AVX and AVX2 are at the thermal design power. One interesting observation is that, as shown in Table 8 , on average, SSE and AVX run at lower frequencies compared to the values presented in Table 2 for HPL and SVD_Band. This is the behavior described by Intel when processors reach the thermal design power (7).

Finally for, SVD_Bulge SSE is the most power consuming compared to AVX and AVX2 since it consumes 5.90% more power compared to AVX and AVX2. This is due to frequency which is higher for SSE as shown in Table 8 . AVX512 consumes less power than the other instructions. The frequency impact can also be verified when observing the plots where turboboost is disabled. In this case, the power consumption of SSE is lower than AVX and AVX2, while AVX512 is still below because its frequency is lower when turboboost is disabled.

For HPL, SVD_Band and SVD_Bulge, most of the time AVX2 consumes more than AVX512 even when turboboost is disabled. We believe that this is because of the frequency. In order to validate our assumption, we ran these three applications at 1.9GHz. Note that we focus only on these three applications since on SToRM, all SIMD instructions already run at the same frequency when turboboost is disabled. For AFF3CT, on the other hand, disabling turboboost shows a similar behavior to SToRM where the larger the register, the larger the power consumption. Note also that differences will be observed since running applications with AVX512 instructions will have the frequency vary depending on the CPU load and performed instructions, whereas we fixed the frequency to 1.9GHz so it cannot vary. Figure 5 shows the power consumption of HPL and SVD when running on yeti at 1.9GHz. The figure shows that HPL and SVD_Bulge have the expected behavior observed on nova. For SVD_Band however, AVX2 and AVX512 show roughly the same power consumption.

As a conclusion, for CPU-intensive applications, both SIMD instructions and frequency seem to impact power consumption. On the other hand, for Memory-intensive applications, frequency seems to be the most impacting factor.

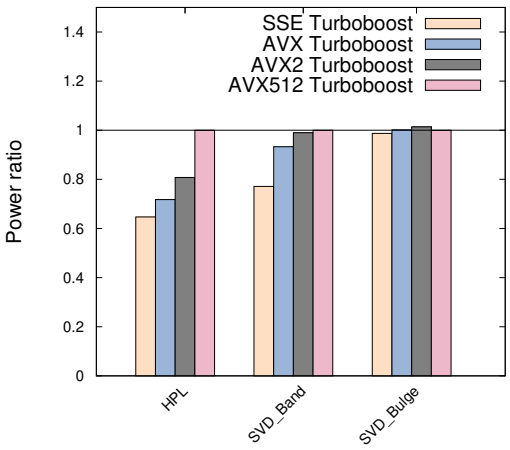

FIGURE 5 Comparison of SIMD instructions power consumption when running at AVX512 frequency (1.9GHz) on yeti for HPL and SVD

4.2 | Impact on performance

In this section, we study the performance behavior of applications. Note that in this section, we will provide explanations on why applications do not have the expected behavior. We will however not explain why an application running using AVX2 is not at least half as slow as running AVX512 for instance. This is because sometimes this problem is due to the algorithm and how the application is designed, which is out of our expertise since we did not design any of these applications.

As expected, on all platforms, the behavior of CPU-intensive applications is similar: the larger the registers size, the better the performance, except for AFF3CT. We will explain AFF3CT's behavior in the end of this section.

On nova and chifflet, all CPU-intensive applications show great performance gain when increasing the register size. For instance, HPL shows a ×3.53 improvement on nova and a ×2.88 on chifflet when using AVX2 compared to SSE. On chifflet, HPL power consumption reaches the thermal design power. As such, its frequency is lowered as shown in Table 7 . The average frequency observed with HPL is the same as the base frequency. This is why the performance when disabling turboboost for AVX2 are so close to the performance with turboboost. Using AVX2 over SSE when disabling turboboost reduces HPL execution time by 71%. This is mainly due to SSE performance which are reduced by a factor of ×2.74 when disabling turboboost.

For SVD_Band and SToRM, disabling turboboost shows an even larger impact when increasing the SIMD registers size. As a matter of fact, using AVX2 over SSE reduces SVD_Band execution time by 65.5% and SToRM execution time by 39.8%.

On yeti however, using AVX512 does not seem to improve performance as much as using AVX2 compared to SSE for all CPU-intensive applications (except AFF3CT). One of reasons lies behind CPU frequency as shown in Table 8 . Moreover, because of TDP, HPL and SVD_Band frequency when using AVX512 is lower than when using AVX2 as stated in Section 4.1. Note that we tried running HPL and SVD on yeti while forcing the frequency to 1.9GHz. This showed a great improvement of the performance ratios of AVX2 over AVX512 for HPL and a small one for SVD_Band (1.47 with the normal behavior and 1.73 when forcing the frequency to 1.9GHz for HPL, 1.11 using the default behavior against 1.13 when the frequency is set to 1.9GHz for SVD_Band). The frequency has also an impact on SToRM performance since disabling turboboost shows a better improvement of AVX512 over AVX2 (12.03% increase using the default configuration and 27.5% improvement when disabling turboboost). This is because for SToRM, as shown in Table 8 , all instructions run at the same frequency when disabling turboboost on yeti.

For SVD_Bulge, SIMD instructions have little to no impact on the performance on nova and chifflet. On yeti however, one can see that there is an impact. We studied the performance (in flop/s) of SVD_Bulge using LIKWID and FLOPS_DP group. The group provides double precision floating point performance in addition to vectorization performance and ratios. Vectorization shows an impact of few GFlops/s on SVD_Bulges while the total performance does not exceed 44 GFlops/s. This means that despite being memory_intensive, this application still computes floating point operations and still uses vectorization which explains why SIMD instructions have an impact on its performance.

AFF3CT execution time on nova shows the expected behavior since AVX2 performs better than SSE. However, the application exhibits the same behavior on chifflet and yeti: the smaller the SIMD instruction registers size, the better the performance. In

other words, for this application, with the parameters that we used, it is better to use SSE rather than AVX2 or AVX512. As stated in Section 3, AFF3CT handles frames which are loaded in the memory. As a consequence, the larger the registers (AVX2 or AVX512), the greater the number of loaded frames. Therefore, for the configuration that we used for AFF3CT, for AVX2 and AVX512, the load exceeds the cache. We compared the ratio of cache misses of the different applications when using SSE, AVX2 and AVX512 on yeti (recall that AVX is not supported by AFF3CT). These results are shown in Table 9 and represent an average over all the cores. Note also that the results show only CPU-intensive applications (thus we exclude SVD_Bulge). Although all applications show a difference when comparing their cache miss ratios between AVX512 and SSE or AVX2, AFF3CT shows the highest difference, especially between SSE and AVX512. As a matter of fact, with AFF3CT, using AVX512 generates more than 14 times more L3 cache misses than using SSE while this ratio is at most 1.80 for the other applications. This explains the performance behavior. Further details are presented in (17).

Application	AVX512/SSE	AVX512/AVX2
HPL	0.83	1.06
SVD_Band	1.71	1
SToRM	1.36	0.98
AFF3CT	14.69	12.64

TABLE 9 : Applications average L3 cache misses over all socket on yeti. The results are presented as a ratio of AVX512 cache misses over SSE or AVX2 cache misses.

4.3 | Impact on applications socket energy consumption

Regarding energy consumption, on all platforms, under the same turboboost configuration,the performance seems to be the major factor impacting energy consumption. This is because SIMD instructions have a large impact on performance compared to power consumption. Thus, except for AFF3CT on chifflet and yeti, for which SSE is better, it is better to use the SIMD instruction with the largest registers.

Regarding frequency, it has more impact on chifflet and yeti since each SIMD instruction runs at a different frequency. For these two platforms, disabling turboboost shows little to no impact on energy consumption.

When looking at the results for each architecture independently, it seems that energy efficiency is the only criteria to target in order to get the best of both worlds: high performance and low consumption. If one assumes that performance and energy are directly linked, a tempting shortcut to save energy consists in optimizing only performance by either always computing faster (21) or sometimes slower (22) in case of imbalanced workloads for small tasks that require the processor to wait for larger tasks. However, our results exhibit examples where this shortcut is not valid. On chifflet (Broadwell architecture), for all applications and SIMD instructions, it is faster to run with turboboost as shown on Figure 3 a. However, with turboboost, it consumes more energy as shown on Figure 3 c. On yeti (Skylake architecture), for all applications and SIMD instructions, it is also faster to run with turboboost as shown on Figure 4 a. Yet, on yeti, unlike on chifflet, it consumes less energy with turboboost as shown on Figure 4 c. As for nova (also Broadwell architecture), for all applications and SIMD instructions, it is also faster to run with turboboost (Figure 2 a), but the consumption with or without turboboost is almost the same (Figure 2 c).

These results illustrate two interesting points:

1. computing faster does not necessarily mean consuming less energy, and although energy consumption is linked to performance, optimizing the latter does not directly translate into an optimization of the former.

2. the best configuration in terms of energy consumption on one architecture (without turboboost on chifflet for instance) can be different of the best configuration for another architecture (with turboboost on yeti) for the exact same application and SIMD instruction. This also means that tuning the configuration (SIMD instruction generation, turboboost or not) for a given application has to be done for each hardware architecture, since the performance in terms of energy efficiency can significantly differ among the architectures.

	application	SSE	AVX	AVX2	AVX512
nova	HPL	3773.65	7906.97	15292.68	-
	SVD_Band	2950	4826.85	8286.38	-
	SVD_Bulge	24458.19	24372.11	24898.53	-
	SToRM	157.72	-	254.67	-
	AFF3CT	2267.4258	-	12713.68	-
chifflet	HPL	7999.30	15602.66	27562.72	-
	SVD_Band	6144.90	9502.15	14995.57	-
	SVD_Bulge	33794.18	33373.57	34208.68	-
	SToRM	442.85	-	552.88	-
	AFF3CT	5458.03	-	19189.8458	-
yeti	HPL	8727.81	16154.19	30789.43	44934.23
	SVD_Band	9330.26	13043.63	19734.75	22697.41
	SVD_Bulge	63064.59	62976.22	63681.07	56818.12
	SToRM	1565.19	-	1102.79	699.03
	AFF3CT	12439.95	-	31832.64	52014.98

TABLE 10 : Mean memory throughput over all socket, in MByte/s, for each SIMD instruction on nova, chifflet and yeti. For SVD_Bulge, only socket 0 memory throughput is presented.

4.4 | Impact on applications DRAM power consumption

Figures 2 e, 3 e and 4 e present the DRAM power consumption on nova, chifflet and yeti.

For all CPU-intensive applications, on all platforms (except SToRM on yeti), the larger the registers size, the higher the DRAM power consumption. In order to explain this behavior, we studied the impact of vectorized instructions on the applications' memory throughput. Table 10 shows the mean memory throughput of all applications on all platforms over all sockets. Note that not all applications show the same memory throughput over all socket. For AFF3CT, the sockets may have different memory throughput. For instance, on chifflet, socket 1 has roughly half the memory throughput compared to socket 0. On yeti the difference is at most the fourth of the maximum observed throughput with SSE. For SToRM however, the difference between the sockets is of one order of magnitude. It is especially the case when using SSE since it varies from 183 to 4400 MByte/s between socket 0 and socket 1 on yeti. For SVD_Bulge, only socket 0 has a large memory throughput compared to the other sockets. As a consequence, for SVD_Bulge, we present only the memory throughput for socket 0.

The results show a strong correlation between the DRAM power consumption and the memory throughput. First of all, one can notice that, for CPU-intensive applications, as the size of the instruction registers increases, the memory throughput increases as well (23) which also leads to larger power consumption. SToRM seems to exhibit a different behavior on yeti where SSE has a larger DRAM power consumption.This behavior is still correlated to memory throughput. Note that for SToRM, the power consumption is low compared to the other applications: it is roughly half the power consumed by the other CPU-intensive applications for AVX2 and AVX512. SToRM throughput (Table 10) also shows the smallest memory movement of the applications compared to the others on all platforms. For instance, on yeti using AVX512, SToRM memory throughput is $\times 0.015$ HPL throughput and $\times 0.03$ SVD_Band throughput. Finally, as stated before, SToRM memory throughput is different between the sockets. However, there is still a correlation between each socket's DRAM power consumption and memory throughput. In other words, for each socket, the larger the memory throughput, the higher is the DRAM power consumption.

Memory-intensive application SVD_Bugle shows almost no impact of SIMD instructions on DRAM power consumption (with a highest ratio of 1.04 between AVX2 and AVX512). Moreover, the first socket has a larger impact on power consumption compared to the other sockets (26.62W for socket 0 and between 10.7 W and 12.2 W for the other three sockets when using SSE).

Finally, turboboost has also an impact on the DRAM power consumption. This is also due to memory throughput which is lower when turboboost is disabled (as there are less requests per unit of time since the processor runs slower).

4.5 | Impact on applications DRAM energy consumption

Figures 2 f, 3 f and 4 f present the DRAM energy consumption on nova, chifflet and yeti.

In order to measure the DRAM energy consumption, we just multiplied the total execution time by the DRAM power consumption. The results show that on all platforms and for all applications, the most performing application is the least energy consuming. Thus, for all applications except AFF3CT, using the SIMD instruction with the largest registers provides the lowest energy consumption.

4.6 | Key findings

This study provided an insight on how thermal design power and SIMD instructions impact performance, power and energy consumption for both processor and memory.

Hardware architectures over the last decades have a clear tendency to increase cores number, CPU frequencies and vectorized instruction size (AVX512 will probably follow the same path over the next processor generations). Despite hitting the dark silicon wall several years ago, this tendency still goes on, and comes with thermal issues that can hinder the HPC applications' performances. It thus becomes crucial to consider thermal design power to efficiently tune the execution of HPC applications.

More specifically, from our observations, one can conclude that:

- Performance and power consumption are more and more related. Because of TDP and SIMD instructions, core frequency may be lowered to a lower value than described in the processor documentation, which directly impacts performance.
- For most CPU-intensive applications (except applications with special design like AFF3CT), the larger the registers size, the better the performance and the energy consumption. Moreover, turboboost has no impact on energy consumption when using the instruction with the largest registers. Thus, if power is more important than performance, one can start with disabling turboboost.
- Frequency has more impact on power and performance of memory-intensive applications than SIMD instructions.
- The larger the SIMD registers size, the larger the power consumption, except for AVX512 since it runs at a lower frequency.
- DRAM power consumption is strongly correlated to memory throughput. Thus, larger SIMD registers sizes lead to larger DRAM power consumption.
- Under the same turboboost configuration, using the SIMD instruction which provides the best performance also provides the best energy consumption for a given architecture.
- Computing faster does not necessarily means consuming less energy for a processor.
- While using turboboost reduces the runtime, it either increases or decreases the energy consumption of a given application for a given SIMD instruction, depending on the processor architecture. More recent architectures take more advantage of turboboost capability as it allows to both reduce the runtime and the energy consumption of a given application on these architectures, while it is not always the case for older architectures.

5 | RELATED WORK

Many studies focused on the energy consumption of vectorization. For instance, Hackenberg *et al* (12) present recent changes in Intel Haswell processors such as AVX frequencies and voltage regulators. Another study (24) compares the time, power consumption and energy consumption of an erasure code running on traditional x86 and more recent SIMD platforms. This paper shows that using SIMD instructions on such applications reduces execution time and energy consumption. Another comparison of Sandy Bridge Processors and Haswell-EP processors shows how the new changes to Intel architectures, like the TDP, challenge the performance analysis (25). The multithreading and vectorization on different processors have been compared (2), and a clear benefit is given to vectorization over multithreading from an energy consumption's perspective. The impact of special instruction sets was also studied, like load and store AVX instruction set (26). Another study provides the energy consumption of different implementations of Gaussian elimination on different architectures (27). While all these studies compare the execution

of applications without SIMD instructions and with a given generation of SIMD instructions, this paper provides a comparison of these instructions' generations (SSE, AVX, AVX2 and AVX512) over a representative range of HPC benchmarks. Moreover, many studies (5, 6) explicitly start their experimental protocol by disabling turboboost capability as it leads to unexpected behavior. On the contrary, in this paper, we provide an in-depth look into turboboost behavior for each generation of SIMD instructions over the aforementioned HPC benchmarks.

Some studies (28, 23) can be considered complementary to our work. The first study (28) presents the impact of SSE and AVX on applications performance. This work was done on older architectures (e.g. Sandy Bridge with 32nm lithography) than the present study (Broadwell and Skylake, with 14nm lithography for both), and thus does not fully reflect the current complexity in dealing with thermal power dissipation. Another similar work (23) compares the execution time, power and energy consumption of an AVX and SSE implementation of a sorting algorithm. It also shows the impact of varying the memory bandwidth on the performance and energy consumption of SIMD instructions. This work is complementary to the present work since it studies the impact of SIMD instructions on a given algorithm and studies other parameters such as memory bandwidth, while we studied the DRAM power consumption for instance. Moreover, we focused on different applications with different profiles. In another paper, an energy model using codelets is provided (29). This paper also examines the energy consumption of scalar and vectorized instructions (using SSE and AVX2). It shows the effect of different profiles on the energy consumption of SSE and AVX2 instructions. It also displays the effect of data located on L2 and L3 caches on the energy consumption of the codelets. Our work is complementary since we study the power and energy consumption of applications for both socket and memory consumption, and the impact of TDP on the performance and power consumption. Finally, different vectorized instructions (SSE and AVX) have been compared while varying the number of threads using different Intel and ARM platforms (30). This paper also evaluates when turboboost improves the energy consumption. This work is the closest to our work since the authors compare vectorized instructions and study the impact of turboboost. However, the architectures they use (Intel Ivy and Sandy Bridge) are much older than in our study, and using SIMD instructions on these old architectures had almost no impact on power, thus leading to conclusions that are not applicable anymore on current architectures. In conclusion, studies from the literature either focus on a given vectorization generation, a given application, old architectures or the energy consumption of sockets only. In this paper, we provide a comprehensive study over four generations of vectorization on recent hardware for representative HPC benchmarks, and our study includes the impact of turboboost.

6 | CONCLUSION

In the race for computing power, manufacturers have stretched the limits of physical architectures, increasing dark silicon use. Through various techniques, such as vectorization and turboboost frequencies, the physical limits of thermal dissipation can be bypassed for some period of time. Yet, these techniques involve a loss of control of the operating system in favor of the hardware drives which, by themselves, regulate the frequency of processors. The complex and low-level trade-off between heat management and computing speed cause unexpected behaviors from a user point of view. Consequently, the performance in terms of runtime and energy consumption of HPC applications becomes difficult to understand. Therefore, in this work, we studied the impact of vectorization and thermal design power on runtime and on processor and DRAM power consumption. For this purpose, we used 3 different architectures and 5 applications with different behaviors. Our conclusions showed that because of thermal design power, performance and power become less and less independent. As a consequence, when trying to understand an application performance, studying its power consumption and frequency can help understanding its behavior. Moreover, our study showed that although using SIMD instructions with larger registers size improves performance and energy consumption, it has a negative impact on both DRAM and processor power consumption. However, AVX512 seems to have a different behavior: its power consumption is lower than the other instructions despite providing better performance. Finally, we showed that turboboost may lead to better performance, but at the cost of higher energy consumption depending on the architecture.

Since power consumption is becoming a major problem, using power capping techniques may provide a good leverage to reduce power consumption. As a consequence, in the future, we plan to study the behavior of the processors when applying a power cap. This will be especially interesting for applications like AFF3CT and SToRM that do not reach TDP. We also plan to study how using hyperthreading in addition to vectorized instructions can impact application behavior.

ACKNOWLEDGMENT

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

We would like to thank Mathieu Faverge for his help on PLASMA. We would also like to thank Laurent Noé for his help on SToRM. Finally, we would like to thank Adrien Cassagne for his help on AFF3CT.

References

[1] TOP500 Supercomputer Site. http://www.top500.org; .

[2] Cebrian JM, Natvig L, Meyer JC. Improving Energy Efficiency through Parallelization and Vectorization on Intel Core i5 and i7 Processors. *Int. Conf for High Performance Computing, Networking Storage and Analysis (SC)* 2012: 675-684. doi: 10.1109/SC.Companion.2012.93

[3] Intel® Xeon® Processor E5 v4 product Family. Thermal Mechanical Specification and Design Guide.; 2017.

[4] Intel® Xeon® Processor E5 v3 product Family. Specification Update.; 2017.

[5] Jakobs T, Reinhardt L, Rünger G. Performance and energy consumption of a Gram–Schmidt process for vector orthogonalization on a processor integrated GPU. *Sustainable Computing: Informatics and Systems* 2020: 100456. doi: https://doi.org/10.1016/j.suscom.2020.100456

[6] Ueno Y, Nakamura R, Kuga Y, Esaki H. Fast Longest Prefix Matching by Exploiting SIMD Instructions. *IEEE Access* 2020; 8: 167027-167041.

[7] Lento G. Optimizing Performance with Intel® Advanced Vector Extensions. Intel White Paper.; 2014.

[8] Balouek D, others . Adding Virtualization Capabilities to the Grid'5000 Testbed. 2013; 367: 3-20. doi: 10.1007/978-3-319-04519-1_1

[9] Intel® Xeon® Processor Scalable Family. Specification Update.; 2018.

[10] Intel® Xeon® Processor E5 v4 product Family. Specification Update.; 2016.

[11] Treibig J, Hager G, Wellein G. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. *International Conference on Parallel Processing (ICPP Workshops)* 2010: 207–216.

[12] Hackenberg D, Schone R, Ilsche T, Molka D, Schuchart J, Geyer R. An Energy Efficiency Feature Survey of the Intel Haswell Processor. *IEEE International Parallel and Distributed Processing Symposium (IPDPS Workshops)* 2015: 896-904. doi: 10.1109/IPDPSW.2015.70

[13] Desrochers S, Paradis C, Weaver VM. A Validation of DRAM RAPL Power Measurements. *International Symposium on Memory Systems (MEMSYS)* 2016: 455–470.

[14] Khan K, Hirki M, Niemi T, Nurminen J, Ou Z. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 2018; 3. doi: 10.1145/3177754

[15] Petitet A, C. Whaley R, Dongarra J, Cleary A. HPL - a Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. *Innovative Computing Laboratory* 2000: Available at http://icl.utk.edu/hpl/.

[16] Haidar A, Kurzak J, Luszczek P. An improved parallel singular value algorithm and its implementation for multicore hardware. *Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)* 2013. doi: 10.1145/2503210.2503292

[17] Cassagne A, Tonnellier T, Leroux C, Gal BL, Aumage O, Barthou D. Beyond Gbps Turbo decoder on multi-core CPUs. *Int. Symp. on Turbo Codes and Iterative Information Processing (ISTC)* 2016: 136-140. doi: 10.1109/ISTC.2016.7593092

[18] Noé L, Gîrdea M, Kucherov G. Seed Design Framework for Mapping SOLiD Reads. *Research in Computational Molecular Biology* 2010: 384–396.

[19] Buttari A, Langou J, Kurzak J, Dongarra J. A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. *Parallel Computing* 2009; 35(1): 38–53.

[20] Cassagne A, Hartmann O, Léonardon M, et al. AFF3CT: A Fast Forward Error Correction Toolbox!. *Elsevier SoftwareX* 2019; 10: 100345. doi: https://doi.org/10.1016/j.softx.2019.100345

[21] Haidar A, Abdelfattah A, Zounon M, et al. The Design of Fast and Energy-Efficient Linear Solvers: On the Potential of Half-Precision Arithmetic and Iterative Refinement Techniques. *International Conference on Computational Science (ICCS)* 2018: 586–600.

[22] Bhalachandra S, Porterfield A, Prins JF. Using Dynamic Duty Cycle Modulation to Improve Energy Efficiency in High Performance Computing. *IEEE International Parallel and Distributed Processing Symposium Workshop* 2015: 911-918. doi: 10.1109/IPDPSW.2015.144

[23] Inoue H. How SIMD width affects energy efficiency: A case study on sorting. *IEEE Symp. in Low-Power and High-Speed Chips (COOL CHIPS)* 2016. doi: 10.1109/CoolChips.2016.7503679

[24] Chen H, Grider G, Inman J, Fields P, Kuehn JA. An empirical study of performance, power consumption, and energy cost of erasure code computing for HPC cloud storage systems. *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)* 2015: 71-80. doi: 10.1109/NAS.2015.7255220

[25] Schuchart J, Hackenberg D, Schöne R, Ilsche T, Nagappan R, Patterson MK. The Shift from Processor Power Consumption to Performance Variations: Fundamental Implications at Scale. *Comput. Sci.* 2016; 31(4): 197–205.

[26] Jakobs T, Rünger G. On the energy consumption of Load/Store AVX instructions. *Federated Conference on Computer Science and Information Systems (FedCSIS)* 2018: 319–327.

[27] Jakobs T, Runger G. Examining Energy Efficiency of Vectorization Techniques Using a Gaussian Elimination. *Int. Conference on High Performance Computing Simulation (HPCS)* 2018: 268-275. doi: 10.1109/HPCS.2018.00054

[28] Gepner P, Gamayunov V, Fraser DL. Early performance evaluation of AVX for HPC. *International Conference on Computational Science (ICCS)* 2011; 4: 452 - 460.

[29] Mazouz A, Wong DC, Kuck D, Jalby W. An Incremental Methodology for Energy Measurement and Modeling. *ACM/SPEC International Conference on Performance Engineering (ICPE)* 2017: 15–26.

[30] Cebrián JM, Natvig L, Meyer JC. Performance and energy impact of parallelization and vectorization techniques in modern microprocessors. *Computing* 2014; 96(12): 1179–1193.

7 | APPENDIX

In this appendix, we provide, for each application, how it was compiled to use the desired SIMD instruction. We also provide the command line that we used for our experiments. If input files are required, we describe how we obtained them (this is especially the case for SToRM).

7.1 | HPL

HPL is compiled against MKL which allows choosing the right SIMD instruction through the environment variable `MKL_ENABLE_INSTRUCTIONS`. It can be set to SSE4_2, AVX, AVX2 and AVX512 (for yeti only).

HPL uses a configuration file (the values that we changed are described in Table 3 a and is simply launched with mpirun: `mpirun -n $nb_cores xhpl`

7.2 | SVD

SVD is also compiled against MKL. Thus setting the desired SIMD instruction is done the same way as HPL.

The command line that we used is the following : `./time_dgesvd_tile -threads=$cores -n_range=$N:$N:1 -nb=$NB -nowarmup -nodyn -nocheck` where $cores is the number of cores that we want to use, and $N and $NB are described in Table 3 b.

7.3 | SToRM data generation and execution command line

SToRM can be compiled using the provided Makefile with the desired SIMD compilation flags (msse4.2, mavx2 and mavx512bw). We chose to run the storm-nucleotide program (and not the storm-color). The command line that we used to run the program is: `./storm-nucleotide-sse42-x-gcc -g data/Homo_sapiens.GRCh38.d na_rm.chromosome.22.fa -r data/test_1000000.fq -N $nb_cores -i 15 -z 180 -t 200 -o /dev/null`

where -g takes a genome file. We downloaded the Homo_sapiens file from https://bioinfo.cristal.univ-lille.fr/yass/data/Homo_sapiens/GRCh38.dna_rm.chromosome.22.fa

-r is the reads file to map against the genome. In order to generate the test_1000000.fq file, we used the sra toolkit (http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.10.0/sratoolkit.2.10.0-ubuntu64.tar.gz). The toolkit provides a command `fastq_dump`. It generates the n first reads (-X option) from an input file (SRR7764388) and writes it to output (-Z).

Note that we used 1000000 on nova and chifflet and 10000000 on yeti.

7.4 | AFF3CT command line

For AFF3CT, we used the compilation guideline provided in the documentation. Just like SToRM, compiling with the desired SIMD instructions is done through compilation flag. Launching the execution is done with `./bin/aff3ct -p 8 -sim-type BFER -sim-cde-type TURBO -m 2.0 -M 2.0 -K 3008 -dec-type TURBO -dec-implem FAST -dec-sub-simd INTER -i 6 -itl-type LTE -dec-sf-type LTE_VEC -dec-sub-max MAX -sim-stats -enc-type AZCW -chn-type NO -n 2000000`.

How cite this article: Amina Guermouche, and Anne-Cécile Orgerie (2021), Thermal design power and vectorized instructions behavior, *Concurrency & Computation: Practice & Experience*.