Postprint of: Diakun, O., Dobrosolski, J., Czarnul, P. (2024). Investigation of Performance and Energy Consumption of Tokenization Algorithms on Multi-core CPUs Under Power Capping. In: Saeed, K., Dvorský, J. (eds) Computer Information Systems and Industrial Management. CISIM 2024. Lecture Notes in Computer Science, vol 14902. Springer, Cham.
https://doi.org/10.1007/978-3-031-71115-2_23

Investigation of performance and energy consumption of tokenization algorithms on multi-core CPUs under power capping

Oksana Diakun[0009−0005−5055−9245], Jan Dobrosolski[0009−0002−8893−3800] and Paweł Czarnul[0000−0002−4918−9196]

Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology, Narutowicza 11/12, 80-233 Poland
pczarnul@eti.pg.edu.pl

Abstract. In this paper we investigate performance-energy optimization of tokenizer algorithm training using power capping. We focus on parallel, multi-threaded implementations of Byte Pair Encoding (BPE), Unigram, WordPiece, and WordLevel run on two systems with different multi-core CPUs: Intel Xeon 6130 and desktop Intel i7-13700K. We analyze execution times and energy consumption for various numbers of threads and various power caps and demonstrate that energy consumption can be minimized for both CPUs, while metrics such as EDP and EDS could be optimized for the i7-13700K CPU. We further show that percentage energy gain versus execution time loss could be optimized by 3-6% and 7-13%, depending on the algorithm, for the two CPUs respectively, by applying proper non-default power caps.

Keywords: tokenization algorithm, power capping, performance-energy optimization, energy consumption, EDP, EDS

1 Introduction

Performance-energy optimization of applications using high performance computing systems has gained much attention in recent years [5]. Most of the contemporary computers feature multi-core CPU(s) and typically at least one accelerator such as a GPU. This applies to all: powerful cluster nodes, workstations, desktop and even mobile machines. Nodes can be additionally interconnected into a cluster, preferably with a low-latency, high-bandwidth network like Infiniband. Parallelization of computations is required for utilization of the full computational power of such systems, at all levels: nodes, compute devices within a node as well as multi-threaded processing for efficient use of each compute device's processing cores. Additionally, techniques such as DVFS and subsequently power capping have emerged as ways for controlling performance-energy profiles of such devices. These can be used for finding interesting performance-energy trade-offs optimizing metrics such as Energy Delay Product (EDP) – energy mulitplied by execution time, Energy Delay Sum (EDS) – weighted sum of energy and execution time, or energy [7]. In this work, we are exploring potential

of power capping for the application of tokenizer training, crucial for natural language processing systems. We are exploring multi-threaded execution using representatives of both server and desktop class CPUs.

The outline of the paper is as follows. Section 2 contains description of state-of-the-art works concerning incorporation of power capping for the purpose of optimizing performance-energy metrics. Energy-aware optimization is presented including for the applications such as training a transformer-based language model, assessment of job's power consumption in the context of NLP. Section 3 outlines the motivations and contribution of this work, along with the descriptions of tokenization algorithms analyzed in this paper: Byte Pair Encoding (BPE), Unigram, WordPiece, and WordLevel. Section 4 discusses the application workflow as well as details of execution of the tokenization algorithms. Section 5 includes descriptions of the experiments that we have executed, including: details of the testbed environments with two different, server and desktop, CPUs; results including energy consumption, execution time, Energy Delay Product (EDP), Energy Delay Sum (EDS) and energy gain versus/relative to time loss meant as the difference between energy gain compared to default power cap and time loss compared to default power cap – percentage wise, all metrics for the power cap imposed on a testbed CPU. Finally, we provide discussion of the results and finalize with a summary and outline of future work in Section 6.

2 Related work

Power capping can be considered at various levels in a parallel system i.e. imposing a power cap on a computing/data center, a particular cluster, a machine or finally processors, memory with individual computing devices such as CPUs[14, 9] and GPUs[8].

Power capping could be enforced using technologies such as DVFS, controlling power states, using dedicated power capping APIs that allow setting upper bounds on the power used by respective compute devices [5]. There are several APIs allowing to set power caps for modern compute devices for various markets: server, desktop and mobile. This is possible for Intel CPUs using Intel RAPL and for NVIDIA GPUs using NVIDIA NVML. Power capping, on one hand, limits the performance of a given compute device, on the other hand it gives potential for lower energy consumption used throughout application execution, in spite of larger execution time. Additionally, metrics that incorporate both execution time and energy consumption, such as energy delay product (EDP) or Energy Delay Sum (EDS), can also benefit from setting non-default power caps.

Energy-aware optimization considering power capping has been demonstrated for numerous configurations and compute devices. Authors of paper [2] maximize performance of a multi-threaded application under a power cap considering two parameters: the number of cores used and the core power state. Performance of numerical algorithms with different computational intensities under power caps are investigated in [4]. Training deep convolutional neural networks for image recognition has been shown to benefit from non-default power caps for optimiza-

tion of energy, EDP and EDS e.g. when optimizing for EDP – 25%–28% of energy was saved with average 4.5%–15.4% performance loss [6]. In [3] authors studied impact of power capping on Intel KNL and KNM on three wordcount-based mini-apps: Map+Shuffle, GroupByKey and ReduceByKey.

There are tools available for automatic determination of particular power caps, optimizing a given metric such as energy, EDP or EDS. This can be done statically (executing an application from start until end) using SPLiT or dynamically using DEPO, for both CPUs [7] and GPUs [8]. In the latter case, both performance and average power under a given power cap are determined dynamically within a short time window, allowing to assess the value of a given metric. The space of power caps is browsed at runtime which allows to find the one optimizing the metric. The tool assumes an initialization phase followed by a tuning phase (searching for a power cap) and the remaining execution phase under the found power cap. DEPO can use one of the two algorithms for browsing the space of power caps: Linear Search (LS) that browses the space of power caps with a predefined step or Golden Section Search (GSS) that minimizes the number of steps searching for the minimum of the metric function under the power cap.

Energy-aware processing of language models has already gained attention. For instance, in [10] authors demonstrated that power-capping on various GPUs can result in optimizing performance-energy trade-offs for NLP applications. Specifically, they trained transformer-based networks including BERT, DistilBERT and Big Bird using NVIDIA V100 GPUs. Applying different power caps, compared to the default 250W limit, resulted in larger percentage energy gains compared to percentage performance losses. For instance, for the aforementioned GPU, using a power cap of 150 W, it was possible to save 12.3% of energy at the cost of 8.5% performance loss for BERT and approx. 15% energy reduction at the cost of less than 10% performance loss for DistilBERT. Interestingly, similar trade-offs were observed for NVIDIA A100, but it was not the case for K80 and T4 GPUs suggesting that such trade-offs depend not only on the application, but also on the testbed computing device(s).

A different but very interesting combination of NLP and AI was presented in [1] for prediction of HPC job's power consumption. The former is used to extract meaningful insights from the job's data. A regression problem for the job's power consumption prediction is solved. The goal of exploration batch size and power limit for optimization of performance-energy trade-offs for training deep neural networks was explored with Zeus in [13]. For DeepSpeech2 trained with LibriSpeech using the NVIDIA V100 GPU, energy consumption–training time Pareto front was presented. A linear combination of Energy To Accuracy and Time To Accuracy is taken into account for optimization. The proposed technique uses a batch size optimizer with Multi-Armed Bandit with Thompson sampling and a just-in-time profiler for obtaining power (NVML is used and 5s measurements for each power limit) and throughput for various power limits during the first epoch. Several models were tested, including DeepSpeech2,

BERT, ResNet-50, ShuffleNet V2 and NeuMF, with reported energy gains of 7%-52% at the cost of training time increase of up to 16%.

In [11] authors proposed EdgeBERT that is an algorithm-hardware co-design aimed at latency-aware energy optimization for multi-task NLP. The solution uses dynamic voltage-frequency scaling (DVFS) for minimal energy consumption while meeting a target latency, at a sentence granularity. The solution requires up to 7x and 2.5x lower energy compared to the standard inference without early stopping and a latency-unbounded early exit method.

3 Motivation and contribution

Based on previous, aforementioned successful attempts in determining nontrivial power caps to optimize energy use, we plan to apply these strategies to tokenizer training. Our goal is to find configurations that optimize performance-energy balance or energy. We will focus on metrics such as Energy-Delay Product (EDP), Energy-Delay Product (EDS), and energy minimization.

What is more, in order to increase the effective coverage of our research, we performed a series of experiments both on a server multi-core CPU as well as a modern multi-core CPU with both performance/efficiency cores, under power capping. Showing results across a spectrum of configurations, we are able to highlight the relevancy of discovered correlations. This information is valuable for developers and researchers with access to server/workstation grade equipment. It is also pertinent to users with desktop-class systems, who are an essential part of rapidly growing community focused machine learning-based natural language processing solutions.

Preparing a well trained tokenizer is a crucial aspect of developing efficient and effective natural language processing systems. Training a tokenizer is essentially a statistical and stochastic process, wherein the primary goal is to break down text into smaller, manageable units called tokens. This process is vital because different corpora, comprising varied linguistic features, necessitate distinct tokenization strategies. For example, languages from different linguistic families may require separate tokenizers to enable localization in large language models (LLMs).

Among the most prominent tokenization algorithms are Byte Pair Encoding (BPE), Unigram, WordPiece, and WordLevel, described in Section 4 in more detail. BPE, initially used in data compression, iteratively merges the most frequent pair of bytes in a corpus. This approach is highly effective in managing subword units, reducing the out-of-vocabulary issue common in language modeling. The Unigram algorithm, on the other hand, starts with a large vocabulary and prunes it down using a language model. It is particularly effective for languages with logographic characters like Chinese. WordPiece, popularized by models like BERT, splits words into a limited set of common subwords, balancing between the character and word levels. This method improves the handling of unknown words and morphological richness of languages. Lastly, the WordLevel algorithm is a straightforward approach where the vocabulary consists of whole

words. While simple, it is less effective for languages with rich morphology or those that do not use whitespace as a word delimiter.

These tokenization methods are the backbone of most state-of-the-art tokenizers, mainly due to their efficiency in capturing linguistic nuances and adapting to different languages and contexts. Their application is crucial in LLMs, as they are required for the models to understand and generate human-like text, making them making them inseparable from current NLP research.

Every research and development endeavor in natural language processing using large language models necessitates the use of a tremendous volume of data. Realistically, even in small-scale experiments, the computational cost of preparing a tokenizer on a new or modified corpus represents a significant expense. The process of tokenization, essential for parsing and understanding text data, requires substantial computational resources, especially when adapting to new languages or unique linguistic features.

Consequently, any percentage reduction in energy consumption during this phase can translate into substantial monetary savings. Furthermore, the environmental impact of such efficiency gains cannot be overstated. Lower energy consumption directly correlates to reduced carbon emissions and a lesser environmental footprint. This aspect is particularly crucial given the growing concerns about the energy-intensive nature of training and deploying LLMs. Therefore, advancements in reducing the energy requirements for tokenizer preparation and other NLP processes contribute not only to economic efficiency, but also to the urgent need for more sustainable practices in the field of artificial intelligence.

Our research provides a novel contribution by optimizing the training process of base tokenizers for energy conservation, a topic not extensively covered in previous literature. Unlike prior studies, which have primarily focused on the accuracy and speed of tokenizers, our work emphasizes the environmental and cost-saving benefits. This approach is especially crucial for those intending to develop new tokenizers using established algorithms mentioned before as a backbone for their solutions. Moreover, the possibilities extend beyond just future developments and novelty applications. By enhancing the efficiency of these base tokenizers, we can provide a ripple effect of energy savings across various applications leading to a positive environmental impact, as well as democratizing the field by lowering the overall cost of development.

4 Application workflow

In order to maximize the impact of the differences between the tested algorithms, we preprocessed the dataset with the Whitespace pretokenizer. This dataset consists of reviews of miscellaneous products and books from online stores. The Whitespace pretokenizer splits text into tokens based on spaces and other whitespace characters. This basic yet crucial step allows for further, more sophisticated tokenization and analysis. Finally, the processed data was saved to a file. With such a procedure we are able to isolate the noise of the additional

computational cost not related with the algorithm that is currently investigated and highlight its energy characteristic.

Every analysis begins with creating an instance of a chosen tokenizer, using an implementation provided via The Huggingface[12] models and trainers from the tokenizers API. The algorithms are prepared using Rust, a language renowned for its performance and concurrency capabilities. Rust's design allows the tokenizers to be highly parallelized, making them ideal for handling large datasets efficiently, which enables our research giving hope for unique powercap configurations that allow for energy consumption optimization using the SPLiT tool discussed in Section 2.

As we delve into the specific tokenization algorithms, it is important to recognize their different levels of complexity and room for efficient parallelization. This section examines used tokenizers, highlighting how their distinct characteristics influence the overall performance of the text analysis pipeline.

Byte Pair Encoding (BPE) starts by treating each word as a sequence of characters with an end-of-word symbol. Initially, each character is a separate token. BPE then merges the most frequent adjacent token pairs repeatedly until a set number of steps or desired vocabulary size is reached. This results in high-frequency character pairs, or byte pairs, which are used as single tokens in further processing. These tokens often represent common character combinations or whole words, allowing the model to efficiently represent and process text data.

The Unigram Language algorithm begins with a large vocabulary of subwords and iteratively prunes it. In each iteration, it calculates the likelihood of the training data under the current vocabulary and then removes each token one at a time. The algorithm measures the change in likelihood from each removal, estimating the loss. Tokens with the least impact on likelihood are pruned. This process continues until the vocabulary reaches the desired size or further pruning no longer optimally reduces the model's complexity.

WordPiece is similar to BPE but differs in its token merging criterion. It starts by segmenting text into characters and builds a vocabulary by combining tokens that minimize loss in a language model's likelihood. This process balances the frequency of individual tokens and their co-occurrences. Unlike BPE, which focuses solely on token frequency, WordPiece better captures linguistic structure. It continues merging the most beneficial token pairs until a specified vocabulary size is reached, creating efficient tokens for the model to process.

WordLevel tokenization is the simplest one of these algorithms. It involves splitting the text into tokens based on spaces and punctuation, treating each unique word as a separate token. This method starts by building a vocabulary of all unique words in the training corpus. Each word is then assigned a unique ID. In processing text, each word is simply replaced by its corresponding ID.

5 Experiments

In our series of experiments, we evaluated the performance of various algorithms across different hardware configurations. Tokenizer fitting initially involves col-

lecting and aggregating statistics into a shared structure. To accurately reflect a real-world scenario and minimize the risk of over-parallelization, the text corpus must be relatively large compared to the available computational resources. If excessive computing power is allocated to a relatively small dataset, there's a high likelihood that the problem will become overly granular. This could shift the computation-communication ratio to a suboptimal level during the tests, potentially distorting the collected results. These concerns have led to the decision to create a small, yet representative dataset that can accommodate multiple testbed platform-algorithm combinations. We will empirically determine a subset of thread numbers and progressively extend it until scaling problems emerge.

After performing numerous tests and refining the dataset, 1,2 and 4 threads configurations were deemed representative for the purpose of portraying how the algorithms scale and behave under the restrain of power capping. The dataset was carefully composed to showcase maximal linguistic richness, capturing as many linguistic features as possible within a size limit of 2GB. This constraint ensures that the duration of a single test run remains below 6 hours. Furthermore, this approach offers promising insights into the scalability of results for larger configurations, provided that the corpus is sufficiently large to avoid impeding the parallelization of the algorithm used.

Following the initial test tuning and analysis, we assigned each machine to perform two series of experiments. This approach was adopted to mitigate the impact of random variations in the results. Each series consisted of 12 tests, each conducted 5 times to eliminate the impact of outlier results, encompassing three different thread configurations for each algorithm. Consequently, this structured methodology led to a total of 180 comprehensive experiments being executed across three distinct hardware configurations.

5.1 Testbed environments

Our testing was conducted on two distinct machines, configured in three different hardware setups.

The first machine, referred to as the 'server machine', operates on Ubuntu 22.04.3 LTS. It boasts 180GB of RAM and is powered by dual Intel(R) Xeon(R) Gold 6130 CPUs. This configuration offers a total of 64 logical processors at 2.10GHz, with the capability to reach a peak frequency of 3.70 GHz. Each of these CPUs has a Thermal Design Power (TDP) of 125W.

The second machine is designed to emulate the typical resources available within a desktop system, also running on Ubuntu 22.04.3 LTS. This setup includes 32GB of RAM and utilizes an Intel(R) Core(TM) i7-13700K CPU, with a TDP equivalent to the Xeon Gold CPUs at 125W.

A unique feature of the Intel(R) Core(TM) i7-13700K CPU enables us to create two distinct sub-configurations, treated as separate entities for our tests. The first sub-configuration uses the CPU's 'performance' cores (cores 0-15), which have a base frequency of 3.40GHz and can reach up to 5.30GHz. The second sub-configuration is limited to the 'efficient' cores (cores 16-23), operating

at a lower maximum frequency of 4.20GHz and, theoretically, consuming less power.

5.2 Results

Efficient cores Due to the fact, that the efficient cores are already heavily optimized, when it comes to energy expenditure - computational efficiency trade-off, all efforts to further optimise it resulted in the lack of positive, or purely negative change in this field. Due to that reason the results will not be presented in detail, manipulating the power caps in that scenario for the considered applications is not advised, based on our experience.

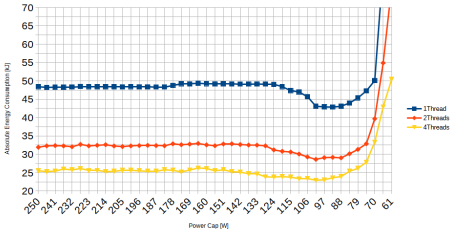

Fig. 1: Absolute Energy Consumption in relation to power cap value for 1, 2 and 4 threads. Intel(R) Xeon(R) Gold, BPE.

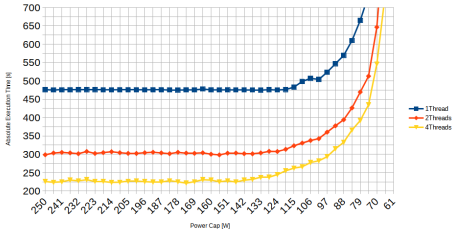

Fig. 2: Absolute Execution Time in relation to power cap value for 1, 2 and 4 threads. Intel(R) Xeon(R) Gold, BPE.

Xeon Gold—BPE algorithm In Figure 1, it is observed that the variant utilizing 4 threads is most efficient in both energy and time metrics. The minimum energy consumption is observed at a power cap of approximately 102 W, amounting to 22.9 kJ. In Figure 2, the absolute execution time for the 4-thread variant exhibits a relatively stable pattern, oscillating around 225 seconds, and begins to increase at a power cap value of 142 W. At the energy optimal powercap value the execution time reaches 282 seconds.

In Figure 3, the value of EDP oscillates around 6 MJs before starting to increase at a power cap of approximately 124 W. The values of EDS for k=1.5 and k=2.0 exhibit similar behavior, maintaining a mostly stable value of approximately 0.97, but they begin to increase at a power cap of 83 W.

The 4 threads scenario, shown in Figure 4, shows a pattern, with energy gain relative to time loss remaining quite close to 0%, achieving the best value of 3% with power cap set to 169 W, and steadily decreasing after the point of 119 W.

Xeon Gold—WPL algorithm In the case of the WPL algorithm, a significantly distinctive minimum can be observed in Figure 5. It occurs for 4 threads,

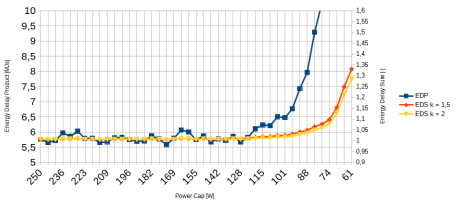

Fig. 3: Energy Delay Product and Energy Delay Sum in relation to power cap value for 4 threads. Intel(R) Xeon(R) Gold, BPE.

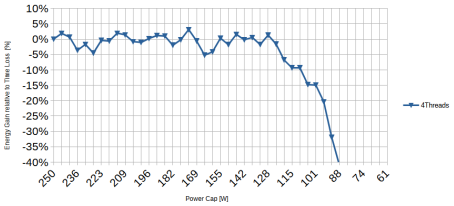

Fig. 4: Energy gain relative to time loss in relation to power cap value for 4 threads. Intel(R) Xeon(R) Gold, BPE.

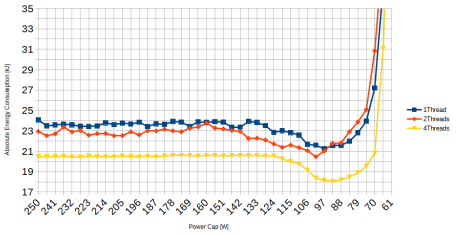

Fig. 5: Absolute Energy Consumption in relation to power cap value for 1, 2 and 4 threads. Intel(R) Xeon(R) Gold, WPL.

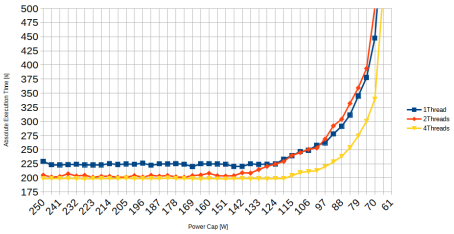

Fig. 6: Absolute Execution Time in relation to power cap value for 1, 2 and 4 threads. Intel(R) Xeon(R) Gold, WPL.

just as in the other cases, and reaches a value of 18.1 kJ, which is the least amongst all of the other cases. However, the nature of its occurrence is similar - at a power cap of 93 W. As for the results shown in Figure 6, the character of execution time is the same as in the previous cases - this value oscillates around 200 seconds until around 115 W, where it then begins to rise. For a power cap of 93 W, the execution time is equal to 228 seconds. It is easily noticeable that this algorithm significantly deviates from its predecessors in terms of achieved values and the distinctiveness of the minimum, yet it still follows the pattern already observed earlier.

In Figure 7, it can be observed that the EDP values follow the same pattern as previously, but this time they appear to be much more correlated with the EDS values. For a power cap of 102 W, a minimum of 3.9 MJs can be observed. Immediately after the minimum, there is a significant increase in values. As for the EDS metrics, both of the analysed cases are oscillating around 1[-] and showing a stable pattern correlated with the shape of EDP until 88 W where it starts to increase smoothly.

When it comes to the values visible in Figure 8 – energy gain relative to time loss, it is interesting to note that for 4 threads the sharp decrease occurs at a far part of the plot, which leads to the conclusion that we can reach a lower power

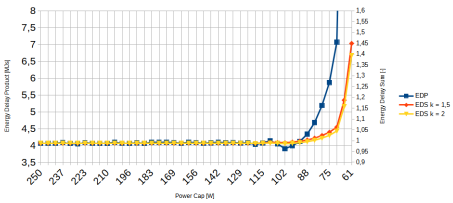

Fig. 7: Energy Delay Product and Energy Delay Sum in relation to power cap value for 4 threads. Intel(R) Xeon(R) Gold, WPL.

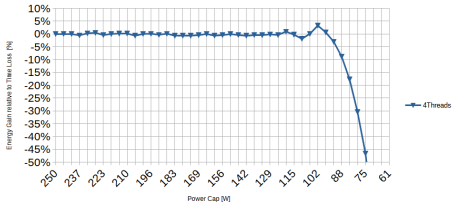

Fig. 8: Energy gain relative to time loss in relation to power cap value for 4 threads. Intel(R) Xeon(R) Gold, WPL.

cap value without significant energy-time tradeoff loss. This graph reaches its maximum for 97 W at 3%.

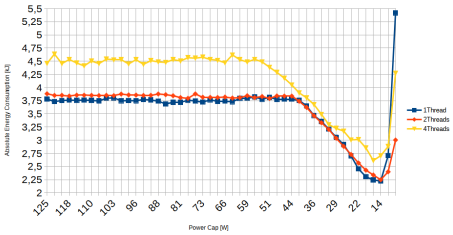

Fig. 9: Absolute Energy Consumption in relation to power cap value for 1, 2 and 4 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, BPE.

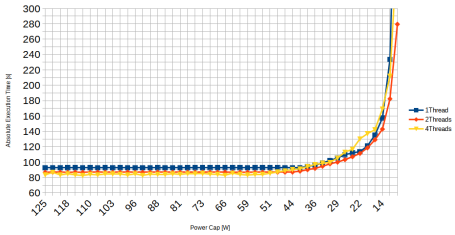

Fig. 10: Absolute Execution Time in relation to power cap value for 1, 2 and 4 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, BPE.

Performance Cores—BPE algorithm Already at first glance, looking at Figure 9, it can be observed that it does not exhibit the same behaviour that could be observed for the results carried out for Xeon Gold. In this case, as far as energy consumption is concerned, the variant that reaches the minimum values in question is the 2-threaded variant. Initially, it fluctuates around the value of 3.8 kJ until the power cap of 43 W is reached, at which point the graph drops dramatically to a value of 2.2 kJ for 16 W. For the execution time, shown in Figure 10, it can be stated that all variants show similar results. The values are consistent, oscillating around 86 seconds, up until around 43 W, at which point the values begin to rise smoothly, reaching a value of 142 seconds for 16 W. Due to the deviation noticeable for the variant with 4 threads, the optimum in this case is analyzed for the scenario of 2 threaded execution.

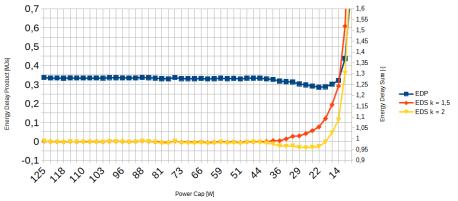

Fig. 11: Energy Delay Product and Energy Delay Sum in relation to power cap value for 2 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, BPE.

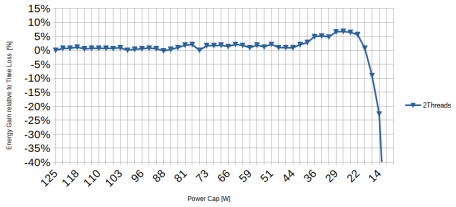

Fig. 12: Energy gain relative to time loss in relation to power cap value for 2 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, BPE.

In Figure 11, the line corresponding to EDP is showcasing high stability with a smooth transition into the easily noticeable minimum, exhibiting different behaviour than in the case of the previously covered Xeon Gold. EDP takes on a constant value of 0.33 MJs until a power cap of 41 W is reached. Further on, the values decrease, reaching a minimum of 0.29 MJs for 22W. The EDS values oscillate steadily around the value of 1 until 41W, then EDS k=1.5 starts to increase and EDS k = 2.0, starts to decrease up until the power cap reaches the value of 27 W and then increases converging to EDS k = 1.5. Figure 12 for the variant with 2 threads is characterized by stable behaviour around the 0% value and reaching a maximum value of 7% at 27W.

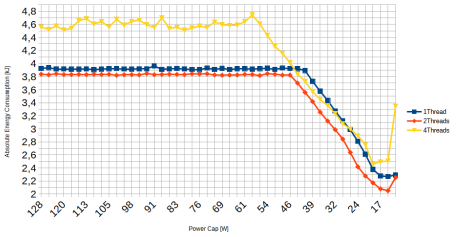

Fig. 13: Absolute Energy Consumption in relation to power cap value for 1, 2 and 4 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, WPL.

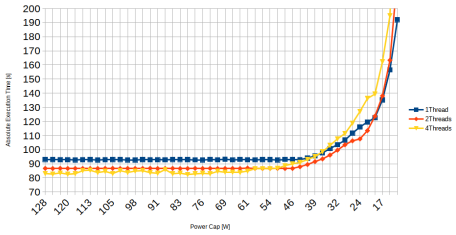

Fig. 14: Absolute Execution Time in relation to power cap value for 1, 2 and 4 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, WPL.

Performance Cores—WPL algorithm In Figure 13 we can observe a significant advantage in energy consumption for the variant with 2 threads. The nature is the same as in previous algorithms for this processor. The 2 threads

result in a constant value of 3.8 kJ until the power cap is reached at 42W, then there is a steep drop to a value of 2 kJ for the power cap of 13 W.

Figure 14 shows the clustered lines for the variants tested. The lowest interspersed lines are for the variants with 2 and 4 threads, for a execution time value of around 87 and 83 seconds respectively, while it is the variant for 2 threads that starts to rise later and for a power cap of 13 W reaches 163 seconds.

An interesting observation can be made while analyzing this particular case, when power caps are applied, the four threaded version shows the biggest energy gains, but at the simultaneous cost of rapid execution time increase. This cannot be stated about the results acquired in the trials that used two threads only, where even though the energy gains were smaller, the final energy consumption - execution time is much more favorable, proving that the tight coupling of gathered metrics is required for proper analysis of the results.

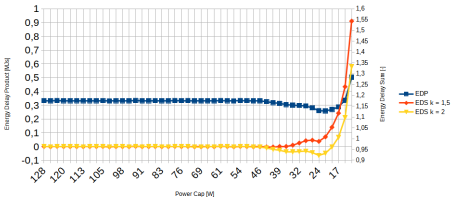

Fig. 15: Energy Delay Product and Energy Delay Sum in relation to power cap value for 2 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, WPL.

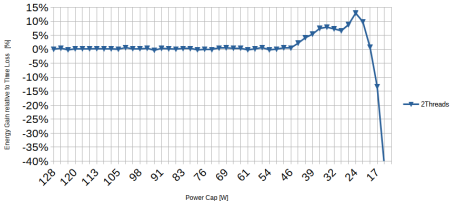

Fig. 16: Energy gain relative to time loss in relation to power cap value for 2 threads. Intel(R) Core(TM) i7-13700K, Performance Cores, WPL.

The line for EDP in Figure 15 oscillates steadily around the constant value of 0.33 MJs, which diminishes at 44 W reaching a minimum for the power cap of 22 W with a value of 0.26 MJs. Behaviour of EDS matches the previous observations, both of the version revolve around the value of 1, until it forks with k=1.5 rising and k=2 reaching a minimum of 0.93 at the power cap of 24 W, before starting to rise and converging with the other one.

Energy gain relative to time loss, shown in Figure 16, reaches a maximum for the variant with 2 threads for a power cap of 22 W and is equal to 13%.

5.3 Discussion of the results

The results presented in Table 1 highlight distinct outcomes when comparing four algorithms tested across the two testbed processors. We can observe consistent patterns in the behavior of these algorithms on both processors, suggesting their reliability and the effectiveness of the energy-limiting tool used in this study. A consistent region of optimal values of the power cap for each processor across

all four algorithms was found: approximately 100W for the Intel Xeon Gold and around 15W for the Performance Cores of the 13th generation i7.

There were notable similarities in metrics such as energy consumption and execution time across tested algorithms. After taking the reduced available resources and limited scale of the problem, results for optimal configurations have been presented. For the Xeon Gold processor, the optimal number of threads for all algorithms was found to be four, whereas for Performance Cores, it was two.

Table 1: Parameters for each architecture

Intel(R) Xeon(R) Gold					
Architecture	Optimal Number of Threads [-]	Minimum Energy Consumption [kJ]	Corresponding Power Cap [W]	Execution Time [s]	Energy Gain versus Time Loss [%]
BPE	4	22.9	102	282	3%
UNI	4	30.7	101	377	3%
WPC	4	22.7	102	279	6%
WPL	4	18.1	93	228	3%
Intel(R) Core(TM) i7-13700K, Performance Cores					
BPE	2	2.2	16	142	7%
UNI	2	2.1	13	159	7%
WPC	2	3.4	16	222	11%
WPL	2	2	13	163	13%

The variation in the optimal thread count can be attributed to the differing computational powers of the CPUs used. According to *cpubenchmark.net*, the single-thread performance of the Intel i7 processor is 4,369 MOps/Sec, significantly higher than the Xeon Gold's 2,067 MOps/Sec. This disparity explains the reduced scalability of problems on our limited-size dataset. With more threads, the balance between computation and communication becomes less than ideal. The observed speed-ups are better for the Xeon Gold for a given number of threads, apparently because of the relatively slower core performance and larger computation/communication ratio, compared to the i7 CPU, for a given number of threads. On the other hand, in order to fully benefit from the Xeon Gold CPU, as a representative of a server CPU, the application would need to scale well for a much larger number of cores/threads than the 4 for which it was tested. Such scalability of the application was not the case, unfortunately. For optimal scalability with increased resources, a larger dataset would be required. However, this would change the nature of the task assigned to the processors and affect the comparison between them.

The aforementioned differences also affected energy optimization. For the Xeon Gold, it is possible to optimize energy consumption, but as can be seen in Figures 3, 7, it is not possible to optimize parameters such as EDP and EDS. On the other hand, on Performance Cores of the i7, besides the possibility of energy optimization, there is also potential for optimizing EDP and EDS. The results,

especially after observing how the problem size for these algorithms impacts scalability, give a very optimistic outlook on the potential of applying power capping to real-life scale problems, and potentially making processing vast text corpora more energy efficient resulting in high monetary gains and substantial reduction of carbon footprint caused by rapid AI growth in recent years.

6 Summary and future work

Within the paper, we have demonstrated that it is possible to perform performance-energy optimization of tokenizer training under CPU power capping. Performed analysis consists of several algorithms including Byte Pair Encoding, Unigram, WordPiece, and WordLevel, using two different types of CPUs: server Intel Xeon 6130 and desktop Intel i7-13700K. We have identified the optimal parallel configurations (4 and 2 threads respectively) and demonstrated that energy gains versus execution time losses can be optimized by 3-6% and 7-13% for two different CPUs through appropriate power capping. This represents a significant advancement in understanding how power caps can be leveraged to balance energy efficiency and performance. Additionally, we have shown that efficiency metrics like EDP and EDS can be optimized for the i7-13700K CPU. This research establishes a foundational understanding of how energy efficiency can be systematically improved across different CPU architectures, showing how important it is to gather insights from different platforms.

Our findings suggest two key directions for future research. First, implementing a stream-based data loader could efficiently process large datasets in distributed storage. Second, validating our thesis with larger datasets (50-100GB) could achieve greater gains by shifting the time distribution away from the preprocessing phase.

References

1. Antici, F., Yamamoto, K., Domke, J., Kiziltan, Z.: Augmenting ml-based predictive modelling with nlp to forecast a job's power consumption. In: Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis. p. 1820–1830. SC-W '23, Association for Compufting Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3624062.3624264
2. Conoci, S., Di Sanzo, P., Pellegrini, A., Ciciani, B., Quaglia, F.: On power capping and performance optimization of multithreaded applications. Concurrency and Computation: Practice and Experience **33**(13), e6205 (2021). https://doi.org/10.1002/cpe.6205
3. Davis, J.H., Gao, T., Chandresekaran, S., Taufer, M.: Studying the impact of power capping on mapreduce-based, data-intensive mini-applications on intel knl and knm architectures (2019)
4. Haidar, A., Jagode, H., Vaccaro, P., YarKhan, A., Tomov, S., Dongarra, J.: Investigating power capping toward energy-efficient scientific applications. Concurrency and Computation: Practice and Experience **31**(6), e4485 (2019). https://doi.org/10.1002/cpe.4485, e4485 cpe.4485

5. Kocot, B., Czarnul, P., Proficz, J.: Energy-aware scheduling for high-performance computing systems: A survey. Energies **16**(2) (2023). https://doi.org/10.3390/en16020890
6. Krzywaniak, A., Czarnul, P., Proficz, J.: Gpu power capping for energy-performance trade-offs in training of deep convolutional neural networks for image recognition. In: Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2022. pp. 667–681. Springer International Publishing, Cham (2022)
7. Krzywaniak, A., Czarnul, P., Proficz, J.: Depo: A dynamic energy-performance optimizer tool for automatic power capping for energy efficient high-performance computing. Software: Practice and Experience **52**(12), 2598–2634. https://doi.org/10.1002/spe.3139
8. Krzywaniak, A., Czarnul, P., Proficz, J.: Dynamic gpu power capping with online performance tracing for energy efficient gpu computing using depo tool. Future Generation Computer Systems **145**, 396–414 (2023). https://doi.org/10.1016/j.future.2023.03.041
9. Krzywaniak, A., Proficz, J., Czarnul, P.: Analyzing energy/performance trade-offs with power capping for parallel applications on modern multi and many core processors. In: 2018 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 339–346 (2018)
10. McDonald, J., Li, B., Frey, N., Tiwari, D., Gadepally, V., Samsi, S.: Great power, great responsibility: Recommendations for reducing energy for training language models. In: Findings of the Association for Computational Linguistics: NAACL 2022. Association for Computational Linguistics (2022). https://doi.org/10.18653/v1/2022.findings-naacl.151
11. Tambe, T., Hooper, C., Pentecost, L., Jia, T., Yang, E.Y., Donato, M., Sanh, V., Whatmough, P.N., Rush, A.M., Brooks, D., Wei, G.Y.: Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference (2021)
12. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019), http://arxiv.org/abs/1910.03771
13. You, J., Chung, J.W., Chowdhury, M.: Zeus: Understanding and optimizing gpu energy consumption of dnn training. USENIX NSDI https://par.nsf.gov/biblio/10402187
14. Zhang, H., Hoffmann, H.: Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. In: Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems. p. 545–559. ASPLOS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2872362.2872375