Benchmarking the Performance of Multicore and Hyper-threaded CPUs

Abbas Muhammad Rabiu
Computer Science Department
Federal University, Dutse
Dutse, Nigeria.
mambaskanya@gmail.com

Maryam Ibrahim Mukhtar
Software Engineering Department
Bayero University, Kano
Kano, Nigeria.
mimukhtar.se@buk.edu.ng

Lamido Yahaya
Computer Science Department
Gombe State University,
Gombe, Nigeria
yahayalmd@gsu.edu.ng

***Abstract*—Multi-core CPUs and Hype-threaded CPUs are believed to be the future of microprocessors and concurrent processing. Each of these CPUs can offer performance benefits to concurrent programmers and other software developers. This paper aims to compare the performance of two similar processors namely: A multi-core CPU and a Hype-threaded CPU that can improve the performance of concurrent applications by using physical /virtual core(s) in their CPUs through hyper-threading. System.CurrentTimeMillis() method was used to measure the running of merge sorts algorithms during benchmarking performed on two different machines running multi-core CPU and hyper-threaded CPU respectively. The result shows that the machine running a multi-core CPU outperforms the other machine running a hyper-threaded CPU. It was further revealed that performance improvements of 0.002ms and 0.16 ms were recorded at the beginning and at the end of the sorting process when the array size contained 1,000 and 1,000,000 elements respectively. It was further revealed that running times of merge sort increase with an increase in the size of the input data on both two test machines. It was concluded that the physical cores present in the Multi-core CPU were faster in concurrent executions than the virtual cores generated by the Hyper-threaded CPU. It was also concluded that a Multicore CPU with four physical cores was more powerful than a hyper-threaded CPU with four logical cores. Hence, to drive the benefits from both two CPUs, a Multi-core combined with hyper-threading technology will give greater performance throughput to concurrent software developers.**

Keywords—Hyper-threaded CPU, Microprocessors Multi-core CPU, Performance, Running times.

I. INTRODUCTION

Multi-core processors and Hype-threaded CPUs perform similar functions but with a small difference in their designs. The former employs physical cores while the latter employs logical/virtual cores to carry out processing. They are both useful for ordinary computer users as well as concurrent programmers to improve performance. So many benefits of these hardware technologies have been discussed by different scholars. Initially, evolutionary computation carried out on multicore processor architecture using benchmarking techniques at the operating system level was carried out. The paper also introduced the benefit of using multicore processor architecture in parallel application development as well as communication networks using MPAC [1]. A similar tool was designed to compare different systems on multicore processor systems using different Operating Systems (OS) [2]. Similar research was carried out to measure the overhead costs and to compare the performance of some concurrency tools using two different OS namely: Windows and Linux OS on Multicore platforms [3]. Furthermore, an overview of the multi-core processor was proposed and the evolutions of multi-core processors together with some design problems were also explained. Additionally, Multi-core advantages over single-core were also described [4]. In addition to the advantages, different synchronization mechanisms were explained to reduce the Heisenberg form of a matrix [4, 5, 6]. Similarly, various parallelization techniques were discussed and various software and hardware tools used for building concurrent applications from the sequential one were explained [7]. A comprehensive study of various factors that affect the performance of multi-core processors was carried out in [8]. It was revealed that Traditional Symmetric Multi-processors (SMPs) are capable of exploiting thread-level parallelism [9]. Additionally, Chip Multi-processor (CMP) systems are capable of providing an increased throughput needed by multi-threaded applications thereby reducing the overheads incurred as a result of sharing "misses" in shared-memory multi-processors [8, 9]. To avoid some problems affecting the performance of single-core, processor architects proposed several schemes to integrate more resources on a single chip such as CMP [10]. Conversely, the architectures of AMD and Intel duo-core processors were studied known as "SPEC CPU2006" using benchmarks to measure the performance of AMD and Intel duo-core processors [11]. However, the disadvantage of high processing speed in AMD and Intel duo-core is that more power is consumed and battery life is affected [12]. Despite the disadvantages of the aforementioned multi-processing power explained in [12], description about some challenges and prospects of multi-core technology were described in [13]. Performance evaluation carried out on CPU using processor affinity on multi-threaded applications using the benchmarking method was presented in [14, 15, 16]. Furthermore, an overview of multi-core processors, multi-core processor parallelism, and performance measurement for multi-core Central Processing Units (CPUs) was presented in [14, 16]. It was further explained that Processor affinity also known as CPU pinning or cache affinity is said to enable the binding and unbinding action of multicore processes and a thread to a central processing unit (CPU) or a parallel CPU [16]. This type of processor known as Core i-Series processors that were built using multithreaded technology is seen today in modern laptops and other desktop computers [17]. Multithreading is said to be another form of carrying out simultaneous processing using multicore CPUs. A larger task is divided or split into threads which are then assigned to different processing cores to perform

computations in parallel to save time [18]. On the other hand, Hyper-Threading Technology as explained by Intel is said to be another hardware improvement that enables multiple threads to execute on different cores of CPU. Generating more threads leads to the corresponding increase in performance because more work can be carried out in parallel [18]. To guide the user of the system, a survey on single and multi-core machines was carried out by studying the current market prices of these machines in some of the largest online stores in Nigeria and other parts of the world [19, 20, 21].

Multi-core CPUs are made up of two or more physical cores namely: Dual-core, Quad-core, and Octa-core, etc. whereas the Hyper-threaded CPUs generate additional cores known as virtual cores to carry out tasks execution in parallel thereby improving the performance throughput [22]. Though, much literature discussed multi-core benefits as well as Hype-threaded CPUs as regards hype-threading and task handling to achieve multitasking, however, benchmarking the two CPUs to compare their performances is lacking in the literature. Therefore, to fill in this gap, this paper studied both multi-core CPUs and Hyper-threaded CPUs through an experimental approach. Benchmarking was carried out on the two machines to compare their performances. To do this, the Merge sort algorithm was implemented and tested on both machines during the experiments. The running times of the merge sort algorithm were recorded from the experiment carried out on both machines to determine the best-performing CPU and to draw conclusions about the results. The outcome of this experiment will educate and guide computer users to make the best choices when buying their computers for different purposes based on the underlying CPU in the system. The research will also enable concurrent programmers to make the right choice between the two CPUs when designing and implementing algorithms and other concurrent software applications to optimize the performance of their programs.

II. LITERATURE REVIEW

Evolutionary computation was carried out on multicore processor architecture using benchmarking techniques at the operating system level. The paper also introduced the benefit of using multicore processor architecture as well as communication networks using MPAC. The performance of MPAC was validated based on both Intel and AMD multicore architectures. It was revealed that Multicore processors were designed to consume reasonable power, dissipate less heat, and make use of cache coherence protocols [1]. It was further revealed that to make full utilization of multicore processors, the applications to be developed to run on multicore must be multi-threaded. It was concluded that there were few applications developed on multicore because there were few programmers with the technical know-how that develop parallel programs that could scale well with the increase in the number of processing cores. It was further concluded that it was unclear whether heterogeneous cores were more efficient than homogeneous cores. It was finally concluded that many issues remained unsolved concerning multicore scalability in parallel applications [1]. A tool was designed to compare different systems on multicore processors. Experiments were carried out using different operating systems. Python libraries as well as user-defined functions were also used to perform some image processing and to carry out some mathematical tasks to impose stress on the systems. It was revealed that this method could be used to optimize the operating system performance for a given hardware [2]. Similar research was carried out to measure the overhead costs and to also compare the performance of some concurrency tools using two different OS namely: Windows and Linux OS on Multicore platforms. The experimental approach was used to carry out benchmarking on a machine with 8 processing cores. Measurements of overheads revealed that Linux is the better OS with fewer overheads as compared to Windows. It was further revealed that the method involved less overhead as compared to threads and classes in concurrent software developments [3]. An overview of the multi-core processor was proposed. The evolutions of multi-core processors together with some design problems were explained. Multi-core advantages over single-core were also described [4]. Different parallel algorithms for linear algebra were further described. The design involved using a general matrix and a systematic matrix. Different synchronization mechanisms were also explained to reduce the Heisenberg form of a matrix. The results obtained from the experiment showed that dynamic task allocation could be very effective on the machine used for the benchmarking; high efficiency was obtained with careful design and implementations of parallel algorithms even for relatively small matrices [4, 5, 6]. Various parallelization techniques were discussed, and various software and hardware tools used for building concurrent applications from the sequential one were also explained. Profiling was the method used to measure the effects of parallelism on two parallel machines. The results show that performance improvement could be achieved by careful implementation of some parallelization techniques [7].

A comprehensive study of various factors that affect the performance of multi-core processors was carried out in [8]. It was explained that "since the commercial applications have abundant thread-level parallelism, commercial servers were designed as multiprocessor systems or clusters of multiprocessors to provide sufficient throughput". Additionally, traditional Symmetric Multi-processors (SMPs) are capable of exploiting thread-level parallelism. It was concluded that "SMPs suffer from a performance penalty caused by memory stalls due to cache misses and cache-to-cache transfers, both of which require waiting for long off-chip delays". Other researchers have proved that the performance of commercial applications as well as database applications is generally dominated by sharing misses that would require cache-to-cache transfers [9]. Moreover, the process of integrating more resources on a single chip presented in [8] was becoming more apparent in CMP processor designs by building multiprocessor systems on a single chip. Additionally, Chip Multi-processor (CMP) systems are capable of providing an increased throughput needed by multi-threaded applications thereby reducing the overheads incurred as a result of sharing "misses" in shared-memory multi-

processors. It was further explained that a Chip Multi-processor (CMP) design is mostly composed of two or more processing cores employing private level-one caches while sharing a second-level cache. The results obtained showed that a performance increase was achieved but more power was consumed which affects battery life as a result of higher computational speed and incompatibility with certain software [8, 9]. Therefore, to avoid some problems affecting the performance of single-core, processors, architects proposed several schemes to integrate more resources on a single chip. It was revealed that "chip-level integration of caches, memory controllers, cache coherence hardware and routers can improve performance of online transaction processing workloads by a factor of 1.5" [9]. It was similarly explained that using multi-threading designs would allow multi-core processors to execute several threads simultaneously by making use of thread and processor resources. It was further shown that this approach could improve the performances of database applications as compared to a super-scalar processor applied using similar resources [10]. The architectures of AMD and Intel duo-core processors were studied known as "SPEC CPU2006" using benchmarks to measure the performance of AMD and Intel duo-core processors. The overall execution and throughput time measurements of AMD and Intel duo-core processors were reported and compared. Results obtained showed that the execution time of the "CQ56 Intel Pentium Dual-Core" Processor was found to be 6.62% faster than the "AMD Turion II P520 Dual-Core" Processor while the throughput of the "Intel Pentium Dual-Core Processor" was found to be 1.06 times higher than "AMD Turion (tm) II P520 Dual Core" Processor. Therefore, "Intel Pentium Dual-Core Processors" exhibit better performance probably due to the following architectural features: faster core-to-core communication, dynamic cache sharing between processing cores, and the size of the level 2 cache. However, the disadvantage of processing speed is that more power is consumed and battery life is affected [11, 12].

Despite the disadvantages of the aforementioned multi-processing power, a description of some challenges and prospects of multi-core technology was described in [13] using the benchmarking method. It was focused that on the next decade or so when the full power of high-performance computing and parallel processing becomes more readily available to computer users everywhere, it might be very possible to hold the power of computers with hundreds of execution processing cores in the palm. The result obtained showed that multi-core processors are the future of computing [13]. Performance evaluation carried out on CPU using processor affinity on multi-threaded applications using the benchmarking method. Event counter hardware performance was analyzed to understand its impact on performance. The results obtained using different multi-core machines as well as Linux kernels revealed that 70% of performance improvements can be achieved by applying affinity that fits both platform and application characteristics [9, 13, 14]. Furthermore, an overview of multi-core processors, multi-core processor parallelism, and performance measurement for multi-core Central Processing Units (CPUs) was presented in [14]. Factors affecting the performance of single and multi-core memory, Input/Output bandwidth, and processing core communications were evaluated, and CPU clock speeds and cache coherence were also studied. The result shows that several processing cores could affect CPU performance when the workload is divided between the cores. The conclusion was that different approaches used in multi-core CPU performance analysis were taken depending on the purpose of the study [14, 15]. Additionally, Processor affinity also known as CPU pinning or cache affinity is said to enable the binding and unbinding action of multicore processes and a thread to a central processing unit (CPU) or a parallel CPU, the result is that the processes or the threads can execute only on the selected CPU or range of CPUs rather than any of the CPUs. This is seen as a modification of the old central queuing scheduling algorithm that is associated with a symmetric multi-processing operating system (OS). Thus, each item in the queue is assigned a tag identifying its kin processor. At the time of allocation of resources, each task is then allocated to its kin processor in preference to others running tasks. A good example of processor affinity can always be found in the execution of multiple instances of non-threaded applications, such as some graphics-rendering software [12, 16]. Intel Core i-series are high-speed processors built based on a multithreaded approach. The most common types are the Intel Core i9 processor, Intel Core i7 processor, Intel Core i5 processor, and Intel Core i3 processor respectively. These types of processors are very fast and can be used for gaming purposes because of their outstanding performance. Intel claimed that the Intel Core X-series are unlocked processors that are capable of delivering up to 18 cores for extreme gaming, multitasking, and creative production [17].

Multithreading is said to be another form of carrying out simultaneous processing using multicore CPUs. A larger task is divided or split into threads which are then assigned to different processing cores to perform computations in parallel to save time. Games, depending on their design, can be lightly or heavily threaded, unlike older games that were single threaded in performance. Thus, older game engines were made to run on a single core, and they get their major performance boost from increased clock speeds. Today's game utilizes multiple cores to create complex scenes, they handle other parts of 'draw call' instructions using multithreading. Other instructions were sent from the CPU and received by the GPU concerning the textures, and other objects [18]. On the other hand, Hyper-Threading Technology as explained by Intel is said to be another hardware improvement that enables multiple threads to execute on different cores of CPU. Generating more threads leads to the corresponding increase in performance because more work can be carried out in parallel. Hyper-Threading works in such a way that the CPU generates two "execution contexts" in each available core in such a manner that one physical core appears to work like two "logical cores" that are capable of handling

different threads. Thus, the two logical cores handle tasks much more efficiently than the traditional single-core threaded CPU core. Therefore, by taking advantage of CPU idle time when the core in a single-threaded core would be formerly waiting for executing tasks to finish execution before assigning CPU time to other waiting tasks, the Hyper-threaded approach improves throughput in a CPU [18].

Multi-processors are seen today in modern laptops and other desktop computers. Cost-benefit analysis of single and multi-core machines revealed that single-core machine costs $345.95 (NGN160,000). Similarly, the Dual-core machine costs $497.30 equivalent to NGN230,000.00. Similarly, the Quad-core machine also costs $648.65 (NGN300,000). The octa-core machine costs $999.00 (NGN462,037). These prices are the current market prices obtained from Jumia, Amazon, and Shop-inverse some of the largest online stores in Nigeria and the world respectively as of 11th May 2023 and the prices may vary with time [19, 20, 21, 22].

III. METHODOLOGY

I. Type of Data Structure

The type of data structure used in this paper is the array data structure. Random integer values were generated using the "seed value" to fill the array with the required data sizes. The array size ranges between 1,000 to 1,000,000 sizes.

II. Data Generation

The next thing is to fill the defined array with the suitable data type to carry out merge sorts. A positive integer that is 32-bit in size was selected to fit the defined array size.

III. Function Creation

After defining the data size to be used for merge sorting, the next thing is creating a function that contains the "seed" used to generate random values and the needed size of the array (i.e 1,000 to 1,000,000 elements)

IV. Looping

The next thing is using a loop to fill the index of each array size with required random values. Since this experiment was aimed to be performed on two different machines (i.e. Multicore CPU and hyper-threaded CPU), the same seed values were therefore used to generate random numbers. This enabled the experiment to have the same starting point when dealing with the CurrentTimeMillis () method to measure time in Java.

V. CPU Benchmarking

In this paper, System.currentTimeMillis() was used to carry out experiments and to measure the running times of merge sort algorithms recorded on each machine with different CPUs. In Java, System.currentTimeMillis() returns a difference in the measured time between the current time and midnight of January 1970 UTC in milliseconds (ms).

VI. Analysis of Merge Sort Algorithms

Merge Sort carries out sorting using the divide-and-conquer approach to problem-solving. Thus, a larger problem is divided to form subproblems called tasks. The tasks are then assigned to different physical cores in multi-core or to the available virtual cores generated in a hyper-threaded CPU. The final results are then combined at the end of the sorting process to get the overall results. Merge sort has a time complexity of $O(nlogn)$ in the best case and is performed as follows:

- It divides the input elements into two halves initially,
- Then after the division, merge-sort performs sorting recursively on each half and then merges them to obtain the overall result at the end of the sorting process [23].

VII. Pseudo-code of Merge sort Algorithm

According to ref. [23] Pseudo-code of the Merge sort Algorithm can be written in just five steps as follows:

- Step i. Select the Pivot k and test the condition:
  (arr [], l, k), If k> l, then
- Step ii. The middle element is determined by dividing the given array into two halves,
  Middle element (m) = (l+k)/2
- Step iii: Perform merge-sort on the 1st half as follows:
  Merge-sort (arr, l, m)
- Step iv: Repeat Step iii. on the 2nd half as follows:
  Merge-sort (arr, m+1, k)
- Step v: Merge the results obtained from step iii. & iv. as follows by calling the merge function:
  Call merge function (arr, l, m, k)

VIII. Test Machine Specifications

TABLE I Presents some vital information concerning the two machines used to benchmark the two different Processors. They include properties of the machines and the current market prices of the two machines both in Naira and United States Dollars (USD) as of 19th July, 2023 on Jumia, the largest online store in Nigeria.

TABLE I. PECIFICATIONS OF THE MACHINE USED FOR THE EXPERIMENT

Name	**Manufacturer**	**No Physical/Virtual (Efficient) cores**	**Type of core**	**No. of threads**	**Properties**
Multi-core CPU (Quad-core)	Intel	4	Physical	4	"Hp 15 Intel Pent Quad Core (500GB,4 GB RAM +32GB
Hyper-threaded CPU (Core i3-13305 U)	Intel	4	Logical	6	"Lenovo IDEAPAD 15 Inch - INTEL CORE I3 - 20GB

IV. RESULTS

As shown in TABLE II, array sizes containing 1,00,000 to 1,000,000 elements were used to perform sorting operations on machines having Multi-core CPU and Hyper-threaded CPU respectively. The measurement of running times was carried out in milliseconds (ms).

TABLE II. PERFORMANCE COMPARISON CARRIED OUT ON MULTI-CORE CPU

Array size (n)	**Running time of Merge sort on Multi-core CPU (ms)**	**Running time of Merge sort on Hyper-threaded CPU (ms)**
100,000	0.008	0.010
200,000	0.050	0.080
300,000	0.071	0.091
400,000	0.145	0.260
500,000	0.167	0.240
700,000	0.257	0.310
900,000	0.460	0.650
1,000,000	0.680	0.840

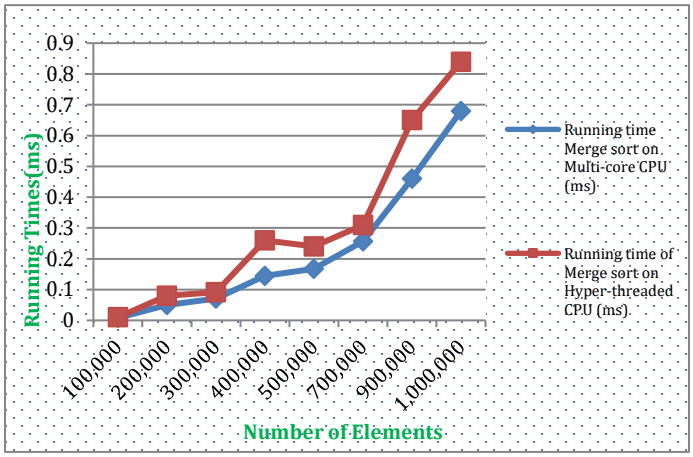

FIGURE I: COMPARISON OF RUNNING TIMES OBTAINED ON BOTH MULTI-CORE AND HYPER-THREADED CPUS

V. DISCUSSION

From TABLE I and FIGURE I, it can be seen that the running times of Merge sort tested on Multi-core CPU and Hyper-threaded CPU were recorded as 0.008 ms and 0.010 ms respectively with the smallest array size. Thus, a performance improvement of 0.002ms was obtained. When the array size increased to 500,000 elements, the running times of 0.167ms and 0.240ms were recorded on Multi-core and Hyper-threaded CPUs respectively. Therefore, a Multicore CPU was found to be 0.073 ms faster than a Hyper-threaded CPU. Similarly, when the array size increases to 1,000,000 elements, running times of 0.680ms and 0.840ms were recorded on both Multi-core and Hyper-threaded CPUs respectively. Thus, a difference of 0.16ms was recorded, meaning that the Multicore CPU was found to be 0.16ms faster than the Hyper-threaded CPU. This revealed that Multi-core CPUs recorded lower running times as compared to Hyper-threaded CPUs right from the beginning to the end of the sorting operations. By recording lower running times, it was shown that Multi-core CPUs outperform Hyper-threaded CPUs throughout the experiment. It was further revealed that running times of merge sort increase with an increase in the size of the input data which follows the finding in ref. [3,15,16]. The good performance of the Multi-core CPU over the Hyper-threaded CPU is because the former has four physical cores while the letter has one logical core that is capable of generating four different virtual cores through hyper-threading. This revealed that the four physical cores present in Multi-core CPUs are generally faster than the four virtual cores generated by the Hyper-threaded CPU during the execution of results.

VI. CONCLUSION

Performances of Multi-core CPU and Hyper-threaded CPU were measured and compared using merge sort algorithms as the software used to test the two underlying hardware (CPUs). From the results and discussion made so far, it can be concluded that the running times of the merge sort algorithm increase with the increase in the size of the array. It can also be concluded that a Multi-core CPU was found to be faster than a Hyper-threaded CPU throughout the experiment. It was further concluded that the physical cores present in the Multi-core CPU were found to be faster in concurrent executions than the virtual cores generated by the Hyper-threaded CPU. Hence, based on the results obtained, it can be concluded that a Multicore CPU with four physical cores was found to be more powerful than a hyper-threaded CPU with four logical cores. Hence, it is recommended that to enjoy the benefits of both two technologies, a Multi-core CPU combined with hyper-threading will give greater performance benefits for both users and concurrent programmers. These new models of CPUs are readily available in the market and their prices are worth the performance benefits drive from them. Other microprocessors with a high number of physical/logical cores can be further tested using other algorithms such as quick sorts and bucket sort to see whether the same or different results can be obtained.

VII. ACKNOWLEDGMENTS

This research work was funded by the Petroleum Development Fund (PTDF) as part of its 2021 Local Scholarship Scheme. The authors highly appreciate the kind the support provided by the fund.

REFERENCES

[1] N. Kasturi, & S.K. Gajendran. "*Performance analysis on Multicore Processors"*. Available: https://sjsu.edu/people/robert.chun/courses/CS247/s4/G.pdf (accessed Mar. 5, 2023)

[2] R. Kannadasan, A. S. Anakath, N. Prabakaran, A. Krishnamoorthy, & S. Ambika. "*Benchmarking of Various Multicore Processors with Different Operating Systems"* [online]. Available:

file:///C:/Users/ABS%20COMO%20SERVICES/Downloads/125986334pdf (accessed Feb. 7, 2023)

[3] A.M. Rabiu, E.J. Garba, Y. B. Baha & Y.M. Malgwi. "Analyzing Overhead Costs on Multi-processing Environments Using Different Operating Systems". *OIRT Journal of Scientific Research.,* Vol. 2, no 5. pp. 44-49. July 2021, doi: https://dx.doi.org/10.53944/ojsr-2220

[4] Balaji, V. "Multi-core processors: An overview". Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, UK. pp.1-6, 2018.

[5] Saleem. S., Ullah, M. I., Lali, M., Nawaz, S. & Nauman, A. "Multi-core program optimization: Parallel sorting algorithms in Intel Cilk Plus". *Inter. J. of Hybrid Infor. Tech.* Vol 7, no 2. pp. 151-164. July 2020., doi http://dx.doi.org/10.14257/ijhit.2014.7.2.15.

[6] Rakhee, C. & Garg, R. B. "Multi-core processor parallelism and their performance analysis". *Inter. J. of Adv. R. in Comp Sci & Tech.* Vol. 2 no. 3. pp. 1-7. 2022.

[7] Leon, A. S. "The ultra-SPARC T1 processor: CMT reliability". *In Proceedings of IEEE* Custom Integrated Circuits Conference. pp. 555–562. Nov., 2016.

[8] Jim, S. "*Streamline processes and reduce processor overhead in CICS"*. Available at: https://techchannel.com/Enterprise/11/2017/processes-processor-overhead-cics (accessed Apr. 2021.)

[9] L. A. Barroso, K. Gharachorloo, A. Nowatzyk & B. Verghese. *"Impact of Chip-Level Integration on Performance of OLTP Workloads"*. Available: http://www.cs.cmu.edu/~agn/Publications/hpca00.pdf. (accessed July 1, 20223)

[10] Vania, M.M., Henrique, C.F., Jean, F.M., & Carlos, A.M. "Evaluating CPU and memory affinity for numerical scientific multithreaded benchmarks on multi-cores". *Int. J. on Comp. Sci. and Inf. Sys,* Vol. 7, no. 1. pp. 79-93. 2013.

[11] Dempsey, P. "Monsters incapacitated multi-core processors". J. of *Eng. & Tech.* Vol. 3 no. 2. pp. 38-41. 2014.

[12] Ananth, G., Anshul G., George K. & Vipin K. "Introduction to parallel computing*"* 2nd Ed.), Addison Wesley, pp. 123-145. 2007.

[13] Anil, S., & Himansh, K. "Multi-core processor technology: Advantages and challenges". *Inter. J. of R. in Eng. and Tech.* Vol. 4, no. 2. pp. 200-201. July, 2019.

[14] IBM. "*Processor affinity and binding"*. Available at: https://www.ibm.com/docs/en/aix/7.2?topic=architecture-processor-affinity-binding. Retrieved 2021-06-08. (Accessed on Jul. 10. 2022.)

[15] A.M. Rabiu, E.J. Garba, Y. B. Baha & Y.M. Malgwi. "Analyzing Overhead Costs on Multi-processing Environments Using Different Operating Systems". [1st International Conference on Computing and Advanced Information Technology (ICCAIT), Nov. 15-17, 2021, ABU Zaria].

[16] A.M. Rabiu, E.J. Garba, Y. B. Baha, Y.M. Malgwi & Mu'azu Dauda. "Performance Optimization of Sorting Algorithms Using Processor Affinity in Java". *OIRT J. of Infor. Tech., Vol. 2, no.* 1, pp. 8-14. 2022. doi: https://doi.org/10.53944/ojit-2207

[17] Intel. "*Intel Core Processor Family"*. Available: https://www.intel.com/content/www/us/en/products/details/processors/core.html. (Accessed on Feb. 6. 2023)

[18] Intel. "What is hyper-threading?" Available: https://www.intel.com/content/www/us/en/gaming/resources/hyper-threading.html, (accessed July 1, 20223)

[19] Shopinverse. *"HP EliteBook Folio 9480m - Intel Core I7 - 500GB HDD - 4GB RAM - Keypad Light*". Available: https://shopinverse.com/products/hp-elitebook-folio-9480m?variant=35161075122338&currency=NGN&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gclid=Cj0KCQjwpPKiBhDvARIsACn-gzC9PNxK-ztDZc57NelexEPwio-6dNGusXtu4Z4Niu7AD5kpax5DNLsaAgNrEALw_wcB. (Accessed on May. 10, 2023).

[20] Jumia. "*HP 6000 SFF Dua Core 8GB RAM, 500GB HDD, Win 10 Pro, MS Office"*. Available: https://www.jumia.com.ng/hp-6000-sff-dua-core-8gb-ram-500gb-hddwin-10-pro-ms-office-35787107.html. (Accessed on May. 10. 2023).

[21] Amazon. "HP victus 15L gaming desktop computer AMD octa-core ryzen 7 5700G processor (Beats i7-10700) 16GB RAM 1TB SSD AMD Radeon RX 6600XT 8GB Graphic (>NVIDIA RTX 3060) USB-C DisplayPort Win11 + HDMI Cable". Available: https://www.amazon.com/HP-Computer-Octa-Core-Processor-DiplayPort/dp/B0BQ389C35?th=1. (Accessed on May 10, 2023).

[22] Geeks for Geeks. "*Merge sorting algorithms"*. Available: https://www.geeksforgeeks.org/merge-sort/ (accessed on Jan. 23, 2023).