Journal, Advances in Mathematical & Computational Sciences
Vol. 11 No. 2, June, 2023
www.isteams.net/mathematics-computationaljournal

Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA

Available online at https://www.isteams.net/ mathematics-computationaljournal.info
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

Development of a Novel Hybrid Sorting Algorithm for Energy Efficiency in Resource-Constrained Devices

[1]**Ayodele, Oluwakemi Sade,** [1]**Owoeye, Sheidu Audu Yakubu &** [2]**Oloruntoba Leke Joseph**
[1]Department of Computer Science, Kogi State Polytechnic, Lokoja, Kogi State, Nigeria
[2]Department of Computer Engineering, Kogi State Polytechnic, Itakpe, Kogi State, Nigeria
Correspondence Email: kemtemmy2009@gmail.com
Phone: +2348069804373

ABSTRACT

In recent years, technological advancement and continuous improvement in computing devices has led to more study in the area of Green Computing with much emphasis on processing time and energy consumption. Intensive studies have been in the area of Energy efficiency of Algorithms with much concentration in improving the hardware. However, there has been little or no work in the area of studying, developing or improving the algorithm itself (energy aware algorithm). The aim of this study is to develop a novel hybrid sorting algorithm (KSU Hybrid derived from Kogi State University where the experiment was conducted) from the quick and insertion sorting algorithms. which took advantage of the strength of both the Quick sort and insertion sort Algorithms. An experiment was carried out to compare the energy consumption of the Quick, merge, insertion and KSU Hybrid. The four sorting algorithms (Quick, merge, insertion and KSU Hybrid) were implemented using three programming languages (C, Java and Python). Time stamp was used to capture the execution time of the sorting algorithm. Power consumed was measured using Joule meter. Use of scripting language has clear drawbacks in terms of energy consumption which should be taken into consideration. KSU hybrid sort is therefore better energy efficient than the Quick, Merge and insertion sort algorithms in most cases. Therefore, this research work provides information on choice of sorting algorithm type and its algorithm implementation style. This is done in order to minimize energy consumption of an algorithm in the current period of explosive growth in the use of smart phones and hand-held devices which run majorly on battery life. This gives developers knowledge on energy efficiency in software leading to choosing some codes over others based on their energy efficiency.

Keywords: Energy Aware, Hand-Held Devices, Green Computing, Hybrid.

Ayodele, O.S., Owoeye, S.A.Y. & Oloruntoba, L.J. (2022): Development of a Novel Hybrid Sorting Algorithm for Energy Efficiency in Resource-Constrained Devices. Journal of Advances in Mathematical & Computational Science. Vol. 10, No. 4. Pp 45-60. dx.doi.org/10.22624/AIMS/MATHS/V11N2P5. Available online at www.isteams.net/mathematics-computationaljournal.

1. INTRODUCTION

In the recent years emphasis has been on computing which focus mainly on developing energy and power efficient devices and the use of non-toxic materials and minimizing e-waste in such design which is more of hardware and less on the software (Christian et al, 2009). A resource-constrained device is device that has limited processing and storage capabilities, and that often runs on batteries. Energy efficiency is using technology that requires less energy to perform the same function. The goal of efficient energy use is to reduce the amount of energy required to provide products and services (Javier, 2018). Therefore, understanding energy usage/ consumption of an algorithm has become a major issue in determining and improving the energy efficiency of any algorithm. Furthermore, many improvements have been introduced in sorting algorithms during the last decade. To facilitate some other operations such as searching, merging and normalization sorting is often required (Waqas, 2016). It is estimated that more than 25% of all computing time is spent on sorting the keys and some installations spending more than 50% of their computing time in sorting files (Essays, UK.,2013).

As a matter of fact much research has been done on the topic of sorting & searching (Deepthi, 2018, Essays, UK., 2013) but there is no single sorting technique which can be considered the best among the rest. (Essays, UK. , 2013). Bubble sort, selection sort and exchange sort are applicable for small input size, insertion sort for medium input size whereas quick sort, merge sort and heap sort are applicable for an application expecting large to huge data size (Author, 2013). All of the above sorting algorithms are comparison based algorithms and hence cannot be faster than $O(n\log_2 n)$, where O and n have their usual meanings (Kazim, 2017; Owoeye, 2016).

For an application, a sorting algorithm is selected according to its computational complexity; ease of implementation and most interestingly recently on its energy efficiency. There is no any single sorting technique which may be called the best among the rest. Bubble sort, insertion sort, selection sort and exchange sort are applicable for input data of small to medium size whereas quick sort, merge sort and heap sort are applicable for an application expecting large to huge data size (Deepthi et al,2018).

Owoeye (2016) improved on the dual pivot Quick sort algorithm, compared the existing Quick sort with modified based on power consumption, energy efficiency, data size and cyclomatic complexity. The empirical study showed that data sizes have effect on power consumption. In (Essays. UK, 2013) a new enhanced sorting algorithm was introduced which shows more efficiency than the insertion sort and other sorting algorithms like bubble sort, quick sort and merge sort.

The technique used for the enhancement in insertion sort is application of improved binary search, adapted from binary search, through which the location of the next element to be placed in the sorted left sub array can be found more quickly than the conventional sequential search used to find that location. In an insertion sort algorithm, there are always two constraints in time complexity. One is shifting the elements and the other one is comparison of the elements. The time complexity is also dependent on the data structure which is used while sorting. If we use array as data structure then shifting takes $O(n^2)$ in the worst case. While using link list data structure, searching takes more time, viz. $O(n^2)$(Kazim, 2017;Surabhi Patel et al., 2014).

Computing devices such as laptops, smartphones, tablets, or other mobile devices, energy consumption is the top priority because they are run on battery, with limited lifespan, as their source of power (kor, 2015). Moreso, most of the energy consumed by theses computing devices is converted into heat, resulting in wear and reduced reliability of hardware components. Also protecting the environment by saving energy and thus reducing carbondioxide emissions is one of today's hottest and most challenging topics (Deepthi, 2018, Bunse C, 2015). From a technological point of view, the realization still falls behind expectations. Moreso, an efficient sorting technique is important to optimize the design of other algorithms that would need sorted key items to work properly and efficiently.

There is no any single sorting technique which may be called the best among the rest. Bubble sort, insertion sort, selection sort and exchange sort are applicable for input data of small to medium size whereas quick sort, merge sort and heap sort are applicable for an application expecting large to huge data size. These sorting algorithms are comparison based and hence can not be faster than *O (n log n).* There are a few algorithms claiming to run in linear time but for specialized case of input data. So, there is an urgent need of a new sorting algorithm which may be implemented for all input data and it may also beat the lower bound (*O (n log n))* of the problem of sorting and also be energy efficient. This work is an effort in that direction.

Insertion sort is an iterative sorting algorithm, a simple in-place sorting algorithm which is relatively efficient for small lists and often adopted to develop sophisticated algorithms though it is highly expensive and inefficient for solving large problems (inefficient for sorting large data). However, previous work pointed out the inefficiency of insertion sorting algorithm in sorting large data sizes. This portends energy challenges for resource-constrained devices. In this research, an energy efficient insertion sort algorithm will be developed by modifying the conventional insertion sort algorithm for computing devices. In this research, an energy efficient insertion sort algorithm will be developed by modifying the conventional insertion sort algorithm for computing devices.

Evaluation of this algorithm will be conducted with some energy efficient sorting algorithms including quick, merge, conventional Insertion and the modified Insertion sorting algorithms using Central Processing Unit (CPU) power utilization, execution time and energy as performance metrics. Proliferation of computing devices which provides quick access to information and fast decision taken is speedily taking over all of human endeavours. Studies in the literature have demonstrated that these computing devices have limited memory and Central processing Unit (CPU) capabilities to process large programs (Niklas et al, 2017) which often accounts for the high time complexity incurred by programs that run on these devices. As a result, there is a need to develop energy efficiency algorithms to manage the complexity of large data sizes on these computing devices since energy efficiency is an essential design criterion for them.

However, past research studies are in the area of hardware implementations which are prone to external influences with little or nothing in the area of Application/Algorithm, they are usually treated as black box. All experimentations from this study are therefore going to be software based. Resource-constrained devices have become an integral part of our life and provide dozens of useful services to their users. However, usability of mobile devices is hindered by battery lifetime.

Moreso, most of the energy consumed by these systems are converted into heat, resulting in wear and reduced reliability of hardware components. Also protecting the environment by saving energy and thus reducing carbon dioxide emissions is one of today's hottest and most challenging topics. From a technological point of view, the realization still falls behind expectations. Therefore, energy conservation and efficiency by applications that run on computing devices is still a big challenge. To this end, developing an energy aware application that can effectively and efficiently be executed remains an open problem. The aim of this research is to develop a novel hybrid sorting algorithm from an improved quick and improved insertion sorting algorithms

2. LITERATURE REVIEW

A number of useful Algorithmic related energy efficient model and work have been reported in literature. This section gives a few of such reported and related work. Deepthi et al (2018) conducted experiments to study how different sorting algorithms have an impact on energy consumption using C language implementation. It was discovered that both time and energy have an impact on the efficiency of these sorting algorithms with quick sort, merge sort and shell sort found to be in the same range of time and energy consumption, followed by insertion and selection sort which is far better than Bubble sort. However, the implementation is only in C language with a non-varying small data size of 10,000. The effect of sorting large data sizes was not considered, varying the algorithm implementation style was also not put into consideration and no modification of algorithm was carried out. Energy models not developed.

Adel Noureddine (2014) presented energy models, approaches and tools that can be used to estimate accurately the energy consumption of software at the code level and the application level. JALEN and JALEN UNIT for estimating how much energy each portion of the codes consumes which helped to provide energy information, draws a model of energy consumption evolution of software based on the value of input parameters. However, this study is not RCD centered and with no application to sorting algorithm. Rivoire et al (2007) proposed an external sorting benchmark for evaluating the energy efficiency of sorting for a wide range of system but focused more on hardware rather than the software domain and also not applicable to resource-constrained devices (RCD).

Bunse et al,(2009) conducted an experiment to show that different sorting algorithms have different energy consumptions and that there is no direct correlation between time complexity of an algorithm and its energy consumption. Owoeye (2016) improved on the dual pivot Quick sort algorithm, compared the existing Quick sort with modified based on power consumption, energy efficiency, data size and cyclomatic complexity. The empirical study showed that data sizes have effect on power consumption. However, the study is not applicable to RCD and no modification of sorting algorithm was carried out.

3. METHODOLOGY

3.1 Experimental Data Acquisition

Source
The data were randomly generated by importing random function from the language(s) libraries.

Size and Format:
In the algorithm execution, five different integer elements were sorted (100,000; 200,000; 300,000; 400, 000 and 500, 000) which were randomly generated.
To reduce measurement error, each data size been considered was executed five times, the average captured and recorded for use.

3.2 Measurement tools

Joulemeter:
Joulemeter is a modeling tool that over the years have been used for measuring the power consumption by servers, virtual machines, desktops, laptops, and software applications running on a computer. The estimates for the power usage of individual components (CPU/Monitor/disk/"base") are provided while running the application. The report consists of power consumption by CPU, monitor, hard disk, and base (base power is the power used by PC even when it is idle). All the numbers are in watts. The total power consumption is also displayed.

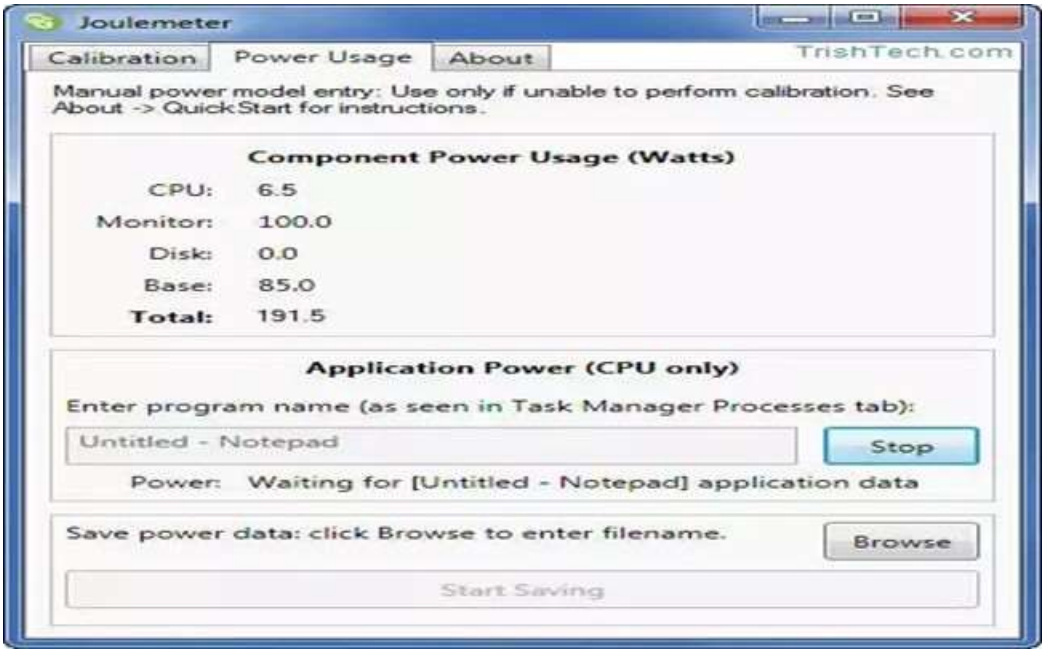

Figure 1: A Joulemeter

TimeStamp

A **timestamp** is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second. A timestamp is the time at which an event is recorded by a computer, not the time of the event itself. In many cases, the difference may be inconsequential: the time at which an event is recorded by a timestamp (e.g., entered into a log file) should be close to the time of the event. This data is usually presented in a consistent format, allowing for easy comparison of two different records and tracking progress over time; the practice of recording timestamps in a consistent manner along with the actual data is called **timestamping.**

3.3 Algorithm Implementation Languages

The three sorting algorithms and the newly developed KSU hybrid sort were run using three different programming languages. This is to:

- ❖ Determine the effect of changing the datasize on energy consumption..
- ❖ Determine how energy is consumed when changing the implementation language.
- ❖ Give a clue as to why energy consumption varies between languages.

Merge, Quick, Conventional Insertion and the newly developed KSU hybrid sorting algorithms were implemented using Java, C and Python Programming Languages. The choice of programming language was categorized based on:

- Virtual machine based languages: Java (compiled and interpreted language)
- Native Languages: C (Compiled language)
- Scripting Languages: Python (Interpreted language)

Two algorithm implementation styles (Iterative and recursive) were used. Recursion is when a statement in a function calls itself repeatedly. The iteration is when a loop repeatedly executes until the controlling condition becomes false. The key difference between recursion and iteration is that recursion is a mechanism to call a function call within the same function while iteration is to execute a set of instructions repeatedly until the given condition is true. Recursion and Iteration are major techniques for developing algorithms and building software applications. The Experiments were conducted on the same computer configuration, and running on Window 7. The sorting algorithms were run on a laptop with the following configuration, HP 630 Notebook PC, 4GB RAM, Intel(R) Core(TM) i3 @2.53 processor.

3.4 Performance Evaluation Metrics

CPU power utilization

This is a measure of Central Processing Unit (CPU) power utilized by Sorting Algorithm. Merge sort, Insertion sort and Quick sort algorithms were implemented using C, Java and Python. Dev C++ and Net beans Integrated Development Environments were used for C and Java programming languages respectively. Python 3.6.0 was used in the implementation of python codes. The sorting algorithms were implemented using recursive and iterative Algorithm implementation styles. Each of the algorithms was implemented in three programming languages, Merge sort and Quick sort algorithms were implemented using the iterative and recursive version for each programming languages.

Data sizes ranges from one hundred thousand (100,000) to five hundred thousand (500,000) at an interval of one hundred thousand. The integers for each data size were randomly generated using the rand() function. The sorting algorithms were placed in functions and classes and called in the main function and class.

Immediately, input data size and click ok to start sorting. The power measurement was stop immediately sorting is completed. The execution time was displayed in millisecond. The start time and end time was used to trace power consumption for the sorting. The experiment is repeated five times for each algorithm using one data size and one programming structure (iterative or recursive). The average of the five experiments was recorded as the value for the data size.

Execution Time

The Execution Time T is the time that it takes for an algorithm to execute. Execution time for sorting was measured using system clock imported from programming langauge's libraries.

```
Start_Time       Invoke _System_clock
     Call_sortingAlgorithm_class/method
       End_Time      Invoke_System_Clock
       Execution_Time = End_Time -  Start_ Time
```

Figure 2: The Algorithm for Execution Time

System.cuurentTimeMillis(), time(Null) and datatime , timeit.default_timer() were time functions used for Java, C and Python Programming Languages respectively. Execution Time was derived by subtracting Start time from end time. Measured times were in seconds.

Energy Formula

Algorithm Energy $Ea = P\ X\ T$

$where\ P = CPU\ Power\ utilization(W) and\ T =\ the\ Execution\ Time(s).$

3.5 Sorting Algorithms

Conventional Insertion Sort Algorithm

Insertion sort is an iterative sorting algorithm. The main idea of this is that at each iteration, insertion sort removes an element, find its ordered position in the sorted array of the previous elements and inserts it there. The algorithm can be written as below:

```
function insertionSortR (array A, int n)
        if n >1
        insertionSortR (A, n-1)
           x = A[n]
           j = n – 1
while j >= 0 and A[j] >x
        A[j+1]        A[j]
j       j – 1
        end while
        A[j+1] = x
        end if
End function
```

Figure 3: Algorithm of the Conventional Insertion Sort

Iterative Quick Sort Algorithm

```
Push the range (0...n) into the stack
Partition the given array with a pivot
Pop the top element.
Push the partitions ( index range ) into a stack if the
range has more than one element
```

Figure 4:Iterative Quick Sort Algorithm

Iterative Merge Sort Algorithm

Push the range (0...n) into the stack
Partition the given array with a pivot
Pop the top element.
Push the partitions (index range) into a stack if the range has more than one element

Figure 5: Iterative Merge Sort Algorithm

Improved Quick Sort Algorithm

Quick sort is very popular since it is the fastest known general sorting algorithm in practice which provides best run-time in average cases while Insertion sort, on the other hand, works very well when the array is partially sorted and also when the array size is not too large. Using the partition approach of the quicksort algorithm the sorting procedure is commenced until we get to a sub arrays whose size is less than a given cut-off-size which distinguishes between the small and large arrays. At the end of this process, we have an array constituting of sub-arrays of sizes less than or equal to the cut-off size that are not sorted themselves but as a whole they are sorted.

```
QUICKSORT (A, p, r)

If p<r
q= PARTITION (A, p ,r)
QUICKSORT(A, p, q)
QUICKSORT(A, q + 1, r)

PARTITION (A, p, r)
x = A[p]
i = p – 1
j = r + 1
while TRUE

   repeat
      j =  j -1
   Until A[j] <= x
   repeat
      i = i +1
   Until A[i] >= x
```

Figure 6: Improved Quicksort using first element as pivot

Modified Insertion Sort

The next procedure was that insertion sort was performed over the entire array to get a sorted result. In the process of running the insertion, binary search is used for comparison which from literature will lead to a time complexity of O(nlogn) as compared to $O(n^2)$ in the worst case. In this research, we will try to combine these two algorithms (Quick sort and insertion sort) in other to take advantage of the strength of these two sorting techniques (the speed of quick sort and also the benefit of effectiveness of insertion sort) using binary search for comparison. Afterwards, hybrid sort (KSU hybrid sort) algorithm (combination of insertion and quick), which is optimum in the sense of minimum average run-time since reducing the execution time has a great effect on the total energy consumption of any computing device.

Energy –Efficient KSU Hybrid Sorting Algorithm

Flowchart of the design of Energy-efficient KSU Hybrid Sorting Algorithm

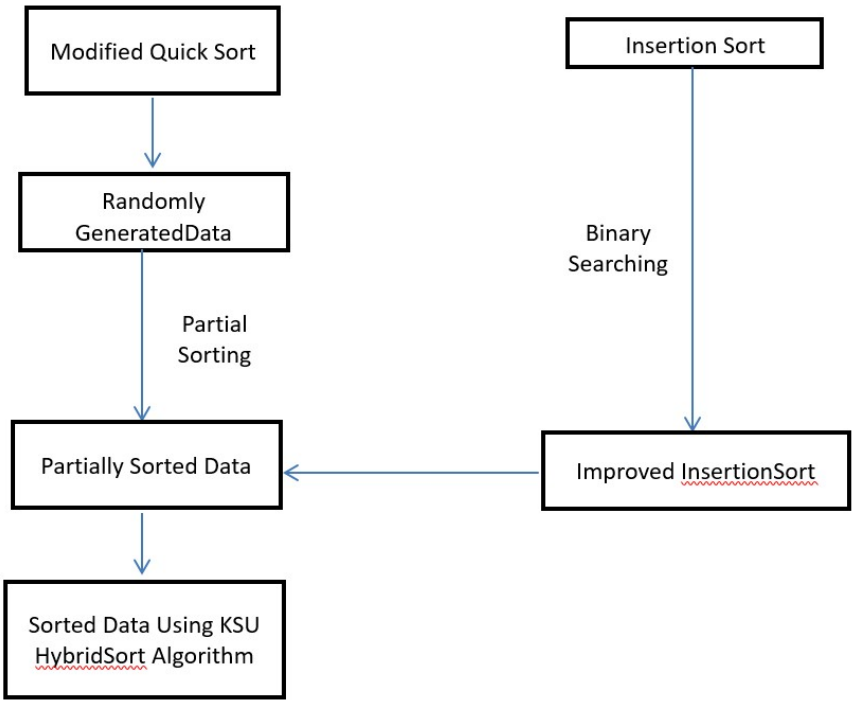

Figure 7: Overview of KSU Hybrid Sorting Algorithm

1. Start
2. Load the Modified Quick sort Algorithm
3. Input: Data Size
4. Enter the number of Parse
5. Partially Sort data
6. Load Modified Insertion Sort
7. Completely Sort the Partially Sorted data
8. Output
   i. Execution Time of sorted data
   ii. Measure CPU power utilized
9. i. Calculate Energy per unit
   ii. Print value of Energy utilized
10. Stop

Figure 8: Program flow of the KSU Hybrid Sort Algorithm

3.6 Metrics Overview
Algorithm Metric Overview

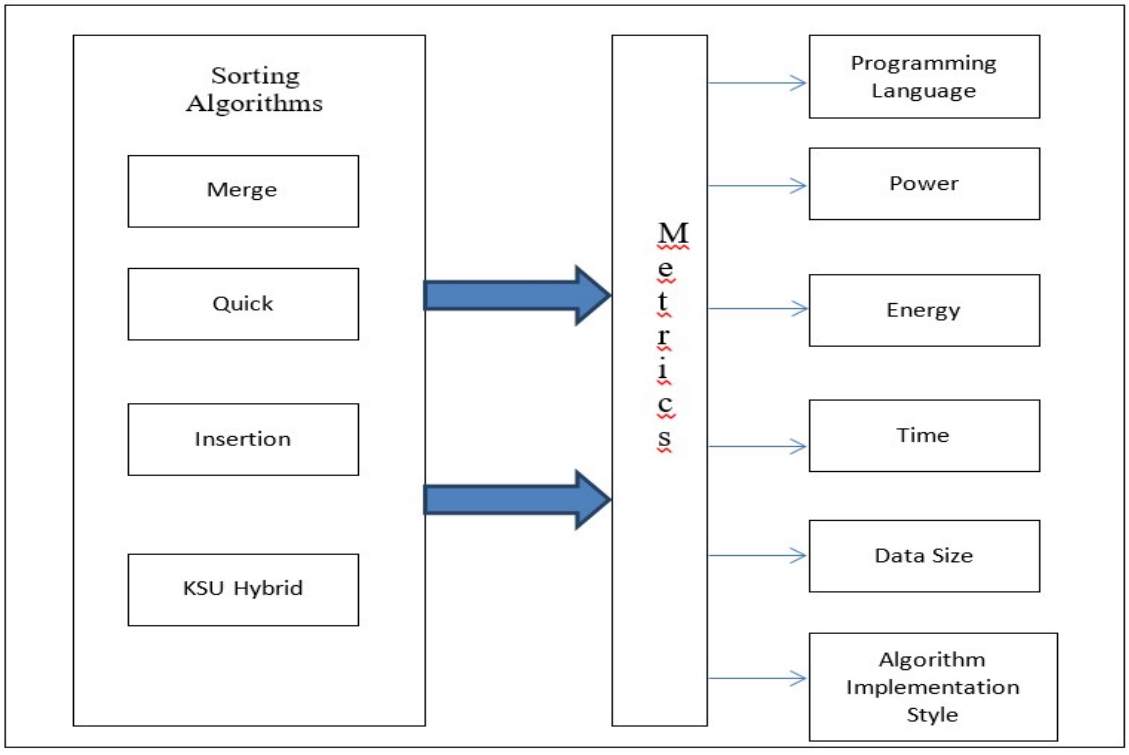

Figure 9: Algorithm Metrics Overview

##Programming Language Metric Based

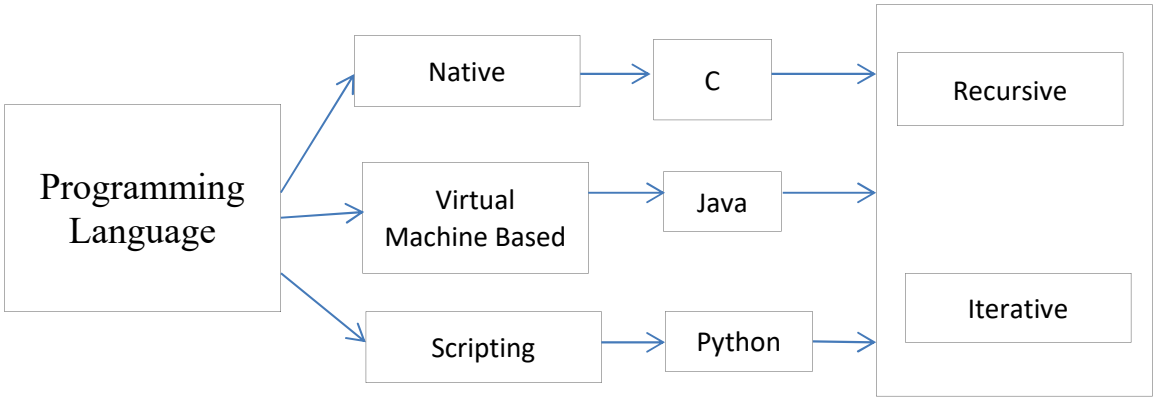

Figure 10: Programming Language Metric Based

3.8 Software Setup

Merge sort, Insertion sort and Quick sort algorithms were implemented using C, Java and Python. Dev C++ and Net beans Integrated Development Environments were used for C and Java programming languages respectively. Python 3.6.0 was used in the implementation. The sorting algorithms were implemented using recursive and iterative styles. Each of the algorithms was implemented in three programming languages, Merge sort and Quick sort algorithms were implemented using the iterative and recursive version for each programming languages. Data sizes ranges from one hundred thousand (100,000) to five hundred thousand (500,000) at an interval of one hundred thousand.

The integers for each data size were randomly generated using the rand() function. The sorting algorithms were placed in functions and classes and called in the main function and class. Time stamp was placed directly above the called function containing the sorting algorithm and another time stamp was placed directly below the called sorting function. This is to ensure that the execution time captured is solely for the sorting algorithm. Execution time was derived by subtracting the start time from end time. Power consumed was measured using Joule meter.

To carry out the experiment, launch Joule meter and the IDE (Dev C++ or Net beans). Run the program, a request to input data size was displayed. Click on browse on Joule meter and specify file name to store power consumed by sorting algorithm per millisecond and recorded. Click on the start button on the Joule meter to record power measurement. Immediately, input data size and click ok to start sorting. The power measurement was stop immediately sorting is completed. The execution time was displayed in millisecond. The start time and end time was used to trace power consumption for the sorting. The experiment is repeated five times for each algorithm using one data size and one programming structure (iterative or recursive). The average of the five experiments was recorded as the value for the data size.

4. RESULTS AND DISCUSSION OF RESULTS

In this section, the presentation and discussion of results from the implementations in chapter three were made in order to evaluate the algorithms performance, to make comparison and deductions based on the energy consumption of the four sorting algorithm using the three programming languages.

Table 1: Energy Comparison of Iterative sorting Algorithms Implementations in Java

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in Java						
DATA SIZE (000)	INSERTION	MERGE SORT	QUICK SORT	KSU HYBRID SORT	% Reovition KSU to Insertion	% Reovition KSU to Merge	% Reovition KSU to Quick
100	76.92423296	0.417088	0.38864	0.27918	27454%	49%	39%
200	886.2912867	0.899096	0.717664	0.57002	155384%	58%	26%
300	3942.79201	1.7496	0.9885	0.7785	506360%	125%	27%
400	10900.63841	2.174592	1.365188	1.13637	959151%	91%	20%
500	25093.63495	2.6344	1.65624	1.701492	1474702%	55%	-3%

The energy consumption of the newly developed KSU Hybrid sort is lower than the energy consumed by the insertion, merge and the quick sort implemented in Java Programming language.

Table 2: Energy Comparison of Iterative sorting Algorithms Implementations in C

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in C						
DATA SIZE('000)	INSERTION	MERGESORT	QUICK SORT	KSU HYBRID	% Reovition of KSU to Insertion	% Reovition of KSU to Merge	% Reovition of KSU to Quick
100	124.673909	0.323556	0.143748	0.0968247	128663%	234%	48%
200	564.445199	0.726528	0.2375	0.383905	146927%	89%	-38%
300	1203.68915	1.414728	0.485688	0.671828	179066%	111%	-28%
400	2173.65748	1.791912	0.678912	1.198771	181224%	49%	-43%
500	3360.21043	1.926332	1.32668	1.797643	186823%	7%	-26%

The energy consumption of the newly developed KSU Hybrid sort is lower than the energy consumed by the insertion, merge and the quick sort implemented in C Programming language with low data size but increases as the data size increase compared to other sorting algorithm.

Table 3: Energy Comparison of Iterative sorting Algorithms Implementations in Python

	Energy Consumption(Joule) of three Iterative sorting Algorithms implemented in Python			
Data Size	Insertion	MergeSort	Quick sort	KSU Hybrid Sort
100,000	76723.02136	75.25877822	61.55288501	10.49974
200,000	334925.2484	147.4104068	132.4095482	15.393924
300,000	678011.6974	229.2594413	205.3952788	71.422364
400,000	1219684.322	320.2936903	282.8371916	134.315792
500,000	2097651.857	448.7777238	335.5805971	192.8384

The energy consumption of the newly developed KSU Hybrid sort is lower than the energy consumed by the insertion, merge and the quick sort implemented in Python Programming language.

Table 4.35Energy Comparison Of KSU hybrid sort Algorithms Implementations In C, Java and Python

Data Size	C	Java	Python
100,000	0.0968247	0.207974	10.49974
200,000	0.383905	0.4473924	15.393924
300,000	0.671828	1.2106564	71.422364
400,000	1.198771	1.81773792	134.315792
500,000	1.797643	1.979624	192.8384

Generally, the Energy consumed by the scripting language (Python) is significantly higher than the virtual based language and also to the native code language. It was noted the high energy consumption of the scripting language which can be attributed to the fact that there is the need to interpret then execute the algorithm. This additional step clearly has a higher value in terms of energy consumption

5. CONCLUSION

Energy management plays a major role in determining the Sorting algorithm to be used during the course of building a system. Energy management has become an important issue in computing environments. Therefore, measuring the energy consumption of software is the first step in order to produce an energy efficient code to complement other hardware and system-based approaches. Developers over the years made significant effort to optimize hardware component in pursuant of an energy efficient device treating the algorithm as a black box. The research work analyzed the energy, time execution and power consumption of four different sorting algorithms, the relative implementation in three distinct languages over an average of five data sizes.

5.1 Contribution to knowledge

The contributions of this research work are a series of experimentations aimed at determining the execution time, power and Energy consumption of four Sorting Algorithms implemented over five randomly generated data sizes that are ran five times to compute its average for Execution Time, Power and energy, in three Programming languages (A native language, C; Virtual machine-based language, Java and Scripting language, Python. From the experiment, insertion sort has the highest energy consumption. An improved sorting algorithm called KSU hybrid sort was developed to take the advantage of the strength of the Quick sort and insertion sort. KSU Sort from experiment has less energy consumption than both quick sort and insertion sort in almost all cases. KSU hybrid sort is therefore better energy efficient than the Quick. Merge and insertion sort algorithms in most cases.
Our research therefore provides the basic information to choose a specific sorting implementation to minimize energy consumption for ener

REFERENCES

1. Adel Noureddine (2014)"Towards a better understanding of the energy consumption of software systems"[cs.SE]. Universite des Sciences et Technologie de Lille I. 2014. [Online] https://tel.archives-ouvertes.fr/tel-00961346v2.Retrieved on 9/9/2016.
2. Christian Bunse, Hagen Hopfner, Essam Mansour and Suman Roychoudhury," (2009) Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments" In Mobile data Management Systems, Services and Middleware, 2009. MDM '09 Tenth International Conference on May 2009, pp 600-607. Retrieved [Online] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.543.8109&rep=rep1&type=pdf on 29/09/2017.
3. Deepthi .T, Birunda Antoinette Mary (2018) "Time and Energy Efficiency: A Comparative Study of Sorting Algorithms Implemented in C" In International Conference on Advancements in Computing Technologies (IJFRCSCE) -ICACT 2018 ISSN: 2454-4248 Volume: 4 Issue: 225–27 Available @ http://www.ijfrcsce.org. Retrieved 20/5/2018
4. Essays, UK. (2013). Modified Insertion Sort Algorithm: Binary Search Technique. Retrieved from https://www.ukessays.com/essays/computer-science/modified-insertion-sort-algorithm-binary-search-1967.php? on 29/08/2018
5. Essays, UK. (2013). Increasing Time Efficiency of Insertion Sort. Retrieved from https://www.ukessays.com/essays/computer-science/increasing-time-efficiency-insertion-6036.php?vref=1 on 29/08/2018
6. Owoeye, F.O(2016). An empirical comparative study of Time, Energy and Cyclomatic complexities of a modified two-pivot quichsort algorithm.Thesis (Msc).University of Ibadan.
7. Javier Mancebo, Hector Omar Arriaga, Félix García, Maria Ángeles Moraga, Ignacio García-Rodríguez de Guzmán, Coral Calero, (2018) "EET: A Device to Support the Measurement of Software Consumption", *Green And Sustainable Software (GREENS) 2018 IEEE/ACM 6th International Workshop on*, pp. 16-22, 2018. Retrieved from https://ieeexplore.ieee.org/document/8449823/ on 12/08/2018.
8. Kazim Ali (2017) "A Comparative Study of well known Sorting Algorithms", In International Journal of Advanced Research in Computer Science.Volume 8, No 1. ISSN No. 0976-5697. [Online] www.ijarcs.info/index.php/Ijarcs/article/download/2903/2886 *retrieved on 19/9/2018*

9. Kor A, C. Pattinson, I. Imam, I. AlSaleemi and O. Omotosho (2015), "Applications, energy consumption, and measurement," In *2015 International Conference on Information and Digital Technologies*, Zilina, 2015, pp. 161-Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7222967&isnumber=7222934 Retrieved on 12/01/2018.
10. Surabhi Patel, Moirangthem Dennis Singh, Chethan Sharma (2014) "Increasing Time Efficiency of Insertion Sort for the Worst Case Scenario" In proceedings of Patel IncreasingTE,retrieved from https://www.semanticscholar.org/paper/Increasing-Time-Efficiency-of-Insertion-Sort-for-Patel-singh/19b98726fd50ee4ddbb4ee2e1f59bea1301e3d16?tab=references on 21/01/2018
11. Niklas, Karvonen; Lara Lorna, Jimenez; Miguel, Gomez;Joakim Nilsson; Kikhia Basel Salah; Josef Hall berg (2017) Classifier Optimized for Resource-constrained Pervasive Systems and Energy-efficiencyIn International Journal of Computational Intelligence Systems. 2017, 10 -1 1272-1279.10.2991/ijcis.10.1.86 [Online] Retrieved on 3/7/2018