The Effect of Parallel Programming Languages on the Performance and Energy Consumption of HPC Applications

Muhammad Aqib
Department of Computer Science
FCIT, King AbdulAziz University
Jeddah, Saudi Arabia

Fadi Fouad Fouz
Department of Computer Science
FCIT, King AbdulAziz University
Jeddah, Saudi Arabia

***Abstract*—Big and complex applications need many resources and long computation time to execute sequentially. In this scenario, all application's processes are handled in sequential fashion even if they are independent of each other. In high-performance computing environment, multiple processors are available to running applications in parallel. So mutually independent blocks of codes could run in parallel. This approach not only increases the efficiency of the system without affecting the results but also saves a significant amount of energy. Many parallel programming models or APIs like Open MPI, Open MP, CUDA, etc. are available to running multiple instructions in parallel. In this paper, the efficiency and energy consumption of two known tasks i.e. matrix multiplication and quicksort are analyzed using different parallel programming models and a multiprocessor machine. The obtained results, which can be generalized, outline the effect of choosing a programming model on the efficiency and energy consumption when running different codes on different machines.**

***Keywords*—power consumption; quicksort; high- performance computing; performance; Open MP; Open MPI; CUDA**

I. Introduction

With the addition of multiple cores, the capability of chips to process multiple instructions simultaneously has increased the performance. High-performance computing provides boost in performance but at some stages, it requires more resources to increase the performance. To provide an optimal solution which could be running efficiently and consumes fewer resources, like energy etc. the performance of the computing system must be analyzed.

Multiple performance analysis tools could be used to test the performance of different software applications [1]. This kind of performance analysis studies help to improve the performance of the software application and to provide an optimal solution. Tools that are utilized for the performance analysis of HPC applications use different approaches for the analysis purposes [2].

In earlier work, performance analysis criteria was based upon the computation of speed, the number of threads generated to perform a task and how the memory was utilized to perform those tasks [3]. When considering HPC architecture, it is supposed that there are a large number of processors that are dedicated to performing the computation tasks. So there is an obvious increase in the consumption of the energy resources as well. So, in addition to the optimization techniques to improve performance, it is also necessary to use energy-aware techniques.

Many optimization techniques could be applied to the code to be running in parallel. For example, loop optimization techniques could be implemented to improve the performance of loops in a code. The use of different programming APIs or architectures like Open MP [4], Open MPI [5], CUDA [6] etc. provides the programmers and application developers with the ability to running different blocks of codes in parallel on CPUs and GPUs. These APIs also provide a mechanism to running the code in parallel using multiple cores in HPC environment.

In this paper Open MP, Open MPI, CUDA were used to perform simple computation tasks i.e. matrix multiplication and to sort using quicksort. Matrix multiplication task is considered as one of the expensive tasks as it involves nested loops and performs multiplication and addition of numbers. In both cases, the code was implemented in C++ to measure the computational time and energy consumption in sequential manner. Then parallel programming API's have been used to get the results while performing the same operations in parallel. Comparing the results obtained implementing different models used in HPC with the results in the sequential mode made it possible to analyze the effect of parallel programming languages on the performance and energy consumption in HPC environment.

The rest of the paper is organized as follows. Section 2 describes the work done by other researchers to analyze the performance of parallel programming models and techniques. Section 3 discusses different tools/models/APIs available for parallel programming. In Section 4, the performance analysis model adopted in this study is presented. Section 5 contains the results obtained using different APIs. Section 6 discusses the results presented in section 5. Finally, conclusion and future work is presented in Section 7.

II. Related Work

Many researchers have provided energy consumption analysis of machines having HPC capabilities. Rejitha *et al.* have analyzed the effect of loop optimization techniques on the use of energy consumed by different techniques [7]. Although they have compared different such techniques but they did not implement them using all available models in HPC

environment. In [8], authors have implemented MPI based solutions using different loop optimization techniques. But, their results are also limited to the use of MPI model.

Freeh *et al.* have directly measured the time and energy with the help of power meters consumed by the AMD-64 nodes [9]. The effects of bottlenecks in the memory and communication in these nodes have been measured. According to them, there is a trade-off between time and energy consumed by HPC applications. i.e. If bottleneck problem arises in any node, it will increase the amount of energy consumed for that application. But this could be reduced by increasing the execution time for that application.

Feng *et al*. have emphasized on the need to characterize the power characteristics of high performance applications to control the energy consumption of future HPC applications [10]. According to them, the operational costs to run an application depends on the characteristics of that application. Even if two applications are running on a system for the same amount of time, the energy consumed by them may differ depending upon their characteristics.

In [12, 13], different techniques to estimate energy consumption in embedded systems have been discussed. Although embedded systems in general are different from high performance systems they have a common case i.e. in both systems energy consumption is a critical issue. So the techniques used for comparison in embedded systems may give an idea on how to estimate the energy consumption in HPC systems.

Enos *et al.* in [14] have provided a mechanism to monitor the energy consumed by CPUs and GPUs installed in a HPC machine. This approach is capable of calculating the power consumption by individual CPUs and GPUs. For this purpose they have used hardware devices and other equipment to monitor the power consumed by the system components. In this paper, a software mechanism has been provided to measure the energy consumption.

A very recent work done by Rashid *et al.* [15] provides an analysis of different sorting algorithms. They have implemented these algorithms on ARM based devices. So this work is basically related to mobile devices. But they have identified some factors which affect the energy consumption in those devices. According to them, algorithm implemented to perform a task and the language used affect the energy consumed by that application.

A model to calculate the energy complexity of different algorithms has been proposed in [16]. Although this is not directly related to high performance computing, it provides a model which deals with the energy consumption and the memory layout which is divided into two layers in this model.

III. Parallel Programming Models and Energy Consumption Analysis Tools

In this section, the parallel programming models and the software tool that were used to get data related to energy consumption analysis have been described. For parallel execution of the code blocks used in the experiments different programming models have been used.

Fig. 1 shows a simple code in C++ language to perform matrix multiplication without any optimization. All the loop instructions in this code run sequentially. Even if this code is run as it is on a multiprocessor machine, it will take the same time to execute.

```
void MultiplyMatrices(int nCount, double **matrixA,
                      double **matrixB, double **matrixC)
{
    int i, j, k ;

    for (i = 0; i < nCount; i++)
    {
        for (j = 0; j < nCount; j++)
        {
            matrixC[i][j]=0;

            for (k = 0; k < nCount; k++)
            {
                matrixC[i][j] +=
                    matrixA[i][k]*matrixB[k][j];
            }
        }
    }
}
```

Fig. 1. Matrix multiplication code in C++ without optimization

Different energy consumption analysis tools have been used by researchers to measure the energy consumed during the execution of the code. In this study, Intel Power Gadget 3.0 [11] have been used for this purpose. It is a power monitoring tool developed by Intel. It supports second generation Intel Core processors to monitor the power consumption in that system. Desktop view of this gadget is show in Fig. 2.

Intel Power Gadget GUI have four different sections that shows different readings. “Package Pwr” section shows the overall power consumption and the average power limit. Current CPU frequency is shown as the “Package Frq”. If GPU is attached with the system, its frequency is shown under the label “GT frq”. The overall system temperature is shown in the section named “Package Temp”. It shows both, the current temperature and the max. temperature limit.

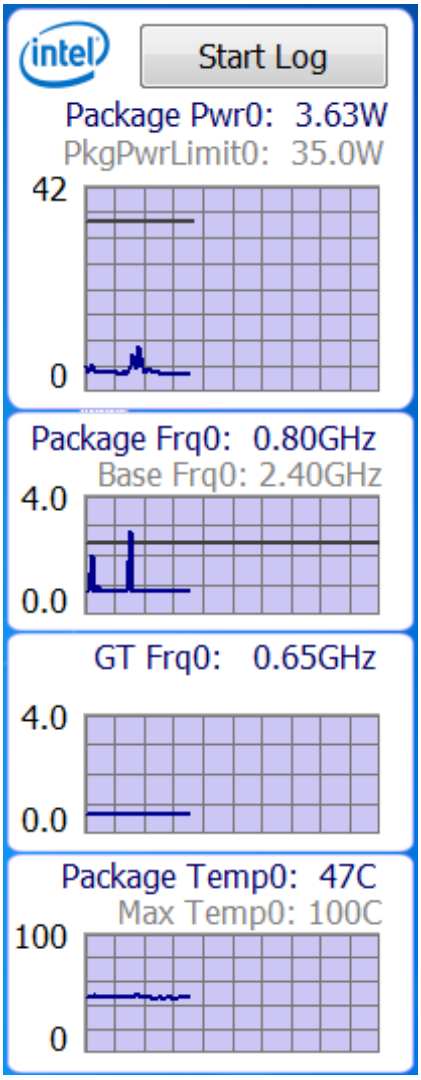

Fig. 2. Intel Power Gadget 3.0

This gadget generates the energy consumption log that provides the power consumption statistics. Log file includes the elapsed timed, package power limit, processor frequency, GT frequency, processor temperature, average and cumulative power of the processor [11]. For the purpose of this study, the "Processor Energy" have been used. This gives the total energy consumed by the processor including the energy consumed by processor cores, GPU, and by other devices.

To run the above code in parallel mode, different parallel programming models have been used. The same code has been implemented using C++ compatible APIs for each parallel programming model. The code has been implemented using Open MP, Open MPI and CUDA. In the following subsections, a brief introduction to these parallel programming models is given.

A. OpenMP

Open MP provides a set of compiler directives. It also includes a set of runtime library routines that are implemented using Fortran, and C/C++. These routines provide support for the parallelism using shared memory model [4].

B. Open MPI

The Message Passing Interface has been implemented in the form of Open MPI [5]. It fully supports the multithreading approach and could be used to develop applications that support concurrent access to memory. It also supports the old versions of MPI like LAM/MPI, LA-MPI and FT-MPI. It also provides options to check the data integrity for processes running in parallel.

C. CUDA

Compute Unified Device Architecture (CUDA) is also a parallel programming model and it is developed by NVIDIA. It runs on a graphical processing unit that supports CUDA. For parallel processing, it provides direct access to the virtual instruction set of GPU [6].

IV. Proposed Model for Energy Consumption Analysis

In this section, the model which was used to perform the analysis and the computing system specifications are presented. To run the programs a multicore hyper threaded machine has been used. The System specifications for that machine are given in the following table.

TABLE I. System Specifications

Component	Name / Capacity
Operating System	Microsoft Windows 7
CPU	Inel® Xeon® CPU E5-2640 @ 2.50 GHz (12 CPUs)
GPU	Nvidia® Tesla K-40
Ram	8 GB
Analysis tool	Intel Power Gadget 3.0

A power consumption analysis model has been proposed. This model describes the process flow and all the steps performed during the analysis process. At the initial stage, before starting the program execution, the energy consumption analysis log needs to be started, and the destination folder for this log file to be selected. After starting the log, the program execution will start. But before starting the multiplication function, the execution time start will be recorded then the multiplication process will be started. After the completion of multiplication process, the time again will be calculated, and both starting and ending times will be written to a separate time log file. Now program will be terminated and the energy consumption analysis log will be stopped. After that, starting and ending time will be available in the time log file and from that time, the energy consumed during that period can be found. A flow chart describing this model is shown in Fig. 3.

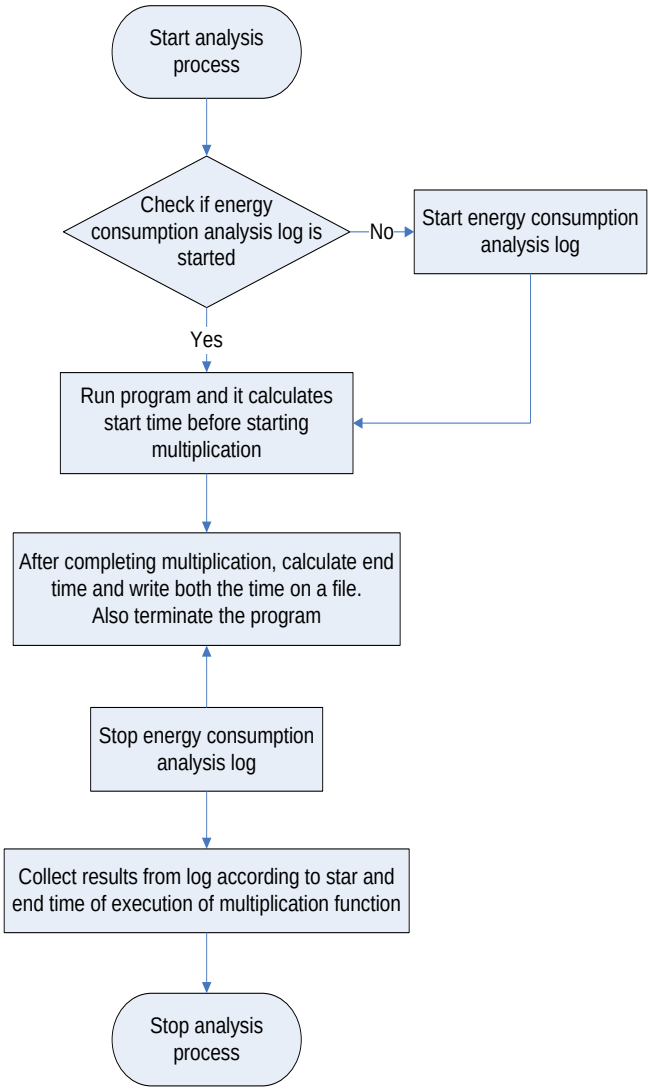

Fig. 3. Proposed model for energy consumption analysis

V. RESULTS

The following section presents the results obtained by running the matrix multiplication program for different matrix sizes and using different programming models. Also the results for running the quick sort algorithm for different array sizes and different programming models are given.

For comparison purposes, different matrix sizes that range from 500 × 500 to 5500 × 5500 have been used. Execution time has been recorded in seconds and the energy log sampling resolution was set to 500ms. This enables the monitoring of energy consumption and other related statistics twice a second. Table 2, and 3, show the results obtained by running each code to multiply square matrixes of five different sizes for each programming model (i.e. C++, Open MPI, Open MP, and CUDA). In table 2, the execution time consumed during the multiplication process is given.

TABLE II. TIME CONSUMED BY DIFFERENT PROGRAMMING MODEL TO MULTIPLY MATRICES OF FIVE DIFFERENT SIZES

Matrix Size	Time Consumption (sec)			
	C++	OpenMP	Open MPI	CUDA
640 × 640	3.042	2.074	1.03	4.055
1280 × 1280	29.062	18.257	17.318	29.408
2560 × 2560	284.131	164.094	181.252	225.279
3840 × 3840	1236.349	571.047	650.066	755.43
5120 × 5120	3101.816	1922.444	1617.374	1789.212

Table 3 shows the results for the energy consumption analysis for the same set of data using the same models for matrix multiplication. Note that, For the purpose of energy consumption analysis, we have measured the overall energy consumed by the system.

TABLE III. ENERGY CONSUMED BY DIFFERENT PROGRAMMING MODEL TO MULTIPLY MATRICES OF FIVE DIFFERENT SIZES

Matrix Size	Energy Consumption (mWh)			
	C++	OpenMP	Open MPI	CUDA
640 × 640	24.051	16.408	8.184	37.497
1280 × 1280	280.361	186.676	158.254	302.709
2560 × 2560	2709.928	1723.636	1689.389	2347.044
3840 × 3840	11677.053	6028.354	6100.8	8019.528
5120 × 5120	29988.794	20762.018	16686.165	19439.64

For quick sort, array sizes have been considered between 128,00,000 to 1,024,00,000. Here it is worth mentioning that for the sorting comparison, array size for CUDA ranges from 12,80,000 to 102,40,000. Similar to matrix multiplication, execution time has been recorded in seconds and energy consumption resolution was also set to 500ms. These results will be discussed in detail in the following section.

TABLE IV. TIME CONSUMED BY DIFFERENT PROGRAMMING MODEL TO SORT ARRAYS OF FIVE DIFFERENT SIZES

Array Size	Time Consumption (sec)			
	C++	OpenMP	Open MPI	CUDA
	× 10^5	**× 10^5**	**× 10^5**	**× 10^4**
128	60.312	112.142	12.012	84.087
256	229.315	431.019	41.058	608.026
512	901.225	1702.979	155.044	649.202
768	2016.177	3956.418	340.095	2133.056
1024	3564.942	6849.089	596.007	5869.163

In table 5, the results obtained by measuring the energy consumed by different programming models to sort the arrays of different sizes have been presented. Same like matrix multiplication, the sampling window was set to 500ms to collect the data for energy consumed by different programming models using the quick sort.

TABLE V. ENERGY CONSUMED BY DIFFERENT PROGRAMMING MODEL TO SORT ARRAYS OF FIVE DIFFERENT SIZES

Array Size	Energy Consumption (mWh)			
	C++	OpenMP	Open MPI	CUDA
	× 10^5	**× 10^5**	**× 10^5**	**× 10^4**
128	551.37	1067.009	94.185	10258.337
256	2740.969	5517.866	1575.13	16745.233
512	11410.318	22433.522	3516.288	24443.189
768	30765.32	60925.113	7831.1	45670.962
1024	64959.75	129887.901	14293.997	103397.188

Figures 4 and 5 show the results obtained by running the matrix multiplication code using the four programming models. Time comparison has been given in Fig. 4, whereas the energy consumption analysis is shown in Fig. 5.

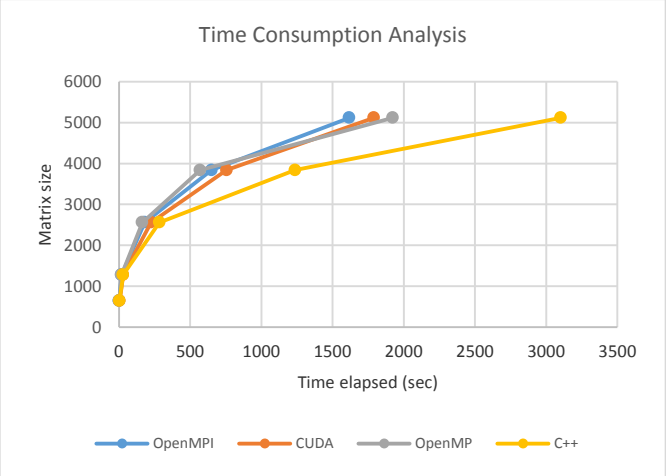

Figure 4. Time efficiency comparison of all four types for matrix multiplication

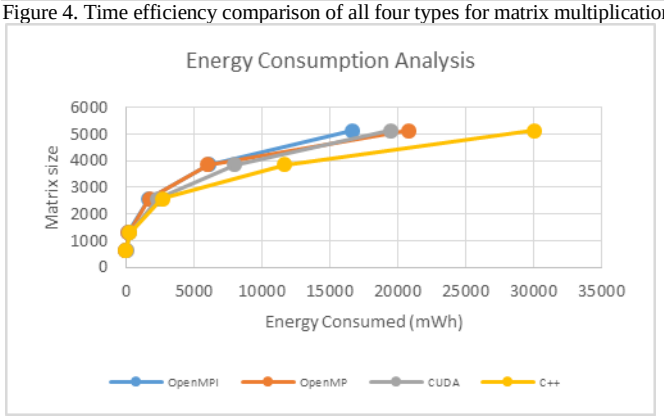

Figure 5. Energy consumed by four models for matrix mulitplication

Figures 6 and 7 show the results obtained by running the quick sort algorithm that is implemented using the four programming models. Time comparison has been given in Fig. 6, whereas the energy consumption analysis is shown in Fig. 7.

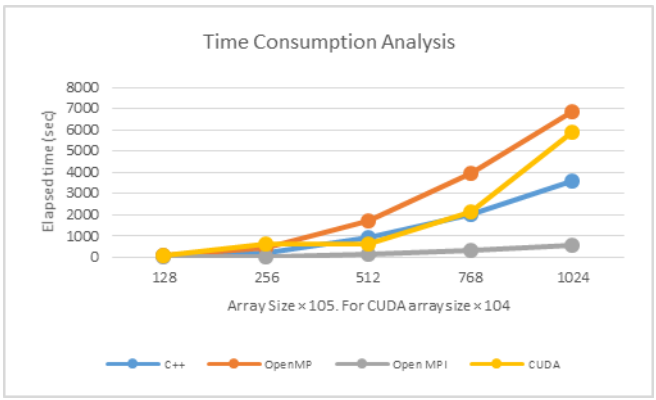

Figure 6. Time efficiency comparison of all four types for quick sort

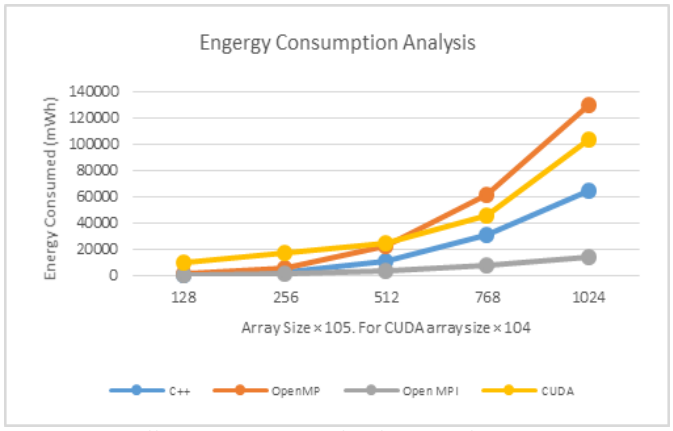

Figure 7. Time efficiency comparison of all four types for quick sort

VI. Discussion

The main purpose of this work is to analyze the performance and energy consumption analysis of different parallel programming models using the computing system and the model described in the previous sections. For this purpose, matrix multiplication and quick sort algorithm have been used. It is obvious that the parallel programming models improve the efficiency and reduce the energy consumption only if there are some blocks of codes that could be parallelized. For example, in matrix multiplication, it is not possible to run all the instructions in parallel, but as the multiplication takes place in the form of rows * columns, so this task could be assigned to multiple threads to run in parallel. Results shown in Fig. 4 and Fig. 5, show that models that support parallel execution of multiple threads produce good results when matrix size is large. For small matrix size, the time and energy consumption is same for all models. And even in some cases, sequential execution is better than the parallel. But when the size increases, the parallel execution produce good results both in terms of time and energy. The results in section 4 show that for large data manipulation, Open MPI performs much better than the other parallel models. On the other hand, results shown in Fig. 6 and Fig. 7 for quick sort show that in most of the cases, sequential execution (C++) produces good results as compared to parallel architectures. Although Open MPI is much more faster than sequential and consumes less energy as compared to sequential execution. But the other two approaches, Open MP and CUDA takes much longer than sequential and in result consumes more energy.

VII. Conclusion and Future Work

Results obtained by running test codes using four models C++ (sequential), Open MPI, Open MP, and CUDA have been discussed in the previous section. The results show that for small calculations, all the models produce the same results in terms of time and energy consumption. Even in some cases as in sorting, the parallel programming models need more resources and time to perform the task. Also, the results obtained by sequential execution are same for small matrix and array sizes. Parallel computation increases performance when running large and complex computations where it is possible to

parallelize the code blocks. Though, every language provides different mechanisms to increase efficiency the default mechanism provided by those models was used. As was mentioned earlier in this paper, the computational tasks of matrix multiplication and sorting were performed on a certain machine. Although the results may differ when performing a different task and utilizing different machine the simple technique used in this work provide a quick and simple way to get a general idea about the performance and energy consumption of a particular programming model on similar machines for different tasks.

In future, this work will be extended by executing some other codes and using different machines or running real applications to get a better estimate of the performance and energy consumption.

REFERENCES

[1] Benedict, Shajulin, et al. "Automatic performance analysis of large scale simulations." Euro-Par 2009–Parallel Processing Workshops. Springer Berlin Heidelberg, 2010.

[2] Wang, Zhiming, et al. "Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter." JCIT 7.1 (2012): 62-70.

[3] Benedict, Shajulin. "Energy-aware performance analysis methodologies for HPC architectures—An exploratory study." Journal of Network and Computer Applications 35.6 (2012): 1709-1719.

[4] Dagum, Leonardo, and Rameshm Enon. "OpenMP: an industry standard API for shared-memory programming." Computational Science & Engineering, IEEE 5.1 (1998): 46-55.

[5] Gabriel, Edgar, et al. "Open MPI: Goals, concept, and design of a next generation MPI implementation." Recent Advances in Parallel Virtual Machine and Message Passing Interface. Springer Berlin Heidelberg, 2004. 97-104.

[6] Kirk, David. "NVIDIA CUDA software and GPU parallel computing architecture." ISMM. Vol. 7. 2007.

[7] Rejitha, R. S., C. Bency Bright, and Shajulin Benedict. "Energy consumption analysis and energy optimization techniques of HPC applications." Energy Efficient Technologies for Sustainability (ICEETS), 2013 International Conference on. IEEE, 2013.

[8] Chowdhuri, Arghyadip, and M. Rajashekhara Babu. "Analysis of Loop Optimization Techniques in Multi-Core Environment using MPI-C." Analysis 2.4 (2011).

[9] Freeh, Vincent W., et al. "Analyzing the energy-time trade-off in high-performance computing applications." Parallel and Distributed Systems, IEEE Transactions on 18.6 (2007): 835-848.

[10] Feng, Xizhou, Rong Ge, and Kirk W. Cameron. "Power and energy profiling of scientific applications on distributed systems." Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. IEEE, 2005.

[11] https://software.intel.com/en-us/articles/intel-power-gadget-20 last accessed on 05-12-2015.

[12] Zotos, Kostas, et al. "Energy complexity of software in embedded systems." arXiv preprint nlin/0505007 (2005).

[13] Castillo, Juan, et al. "Energy Consumption Estimation Technique in Embedded Processors with Stable Power Consumption based on Source-Code Operator Energy Figures." XXII Conference on Design of Circuits and Integrated Systems. 2007.

[14] Enos, Jeremy, et al. "Quantifying the impact of GPUs on performance and energy efficiency in HPC clusters." Green Computing Conference, 2010 International. IEEE, 2010.

[15] Rashid, Mohammad, Luca Ardito, and Marco Torchiano. "Energy Consumption Analysis of Algorithms Implementations." Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on. IEEE, 2015.

[16] Roy, Swapnoneel, Atri Rudra, and Akshat Verma. "An energy complexity model for algorithms." Proceedings of the 4th conference on Innovations in Theoretical Computer Science. ACM, 2013.