Journal of Intelligent Systems and Internet of Things
Vol. 13, No. 02, PP. 129-140, 2024
DOI: https://doi.org/10.54216/JISIoT.130210

Energy saving of cluster computing by CPU frequency Tuning using genetic algorithm

Zainab A. Abdulazeez[1,*], Nihad Abduljalil[2], Ahmed B. M. Fanfakh[3], Ali Kadhum M. Al-Qurabat[4], Esraa H. Alwan[5]

[1] College of Education for Human Sciences, University of Karbala, 56001, Iraq
[2] Department of Air Conditioning and Refrigeration, University of Warith Al-Anbiyaa, Karbala 56001, Iraq
[3] Department of Computer Science, College of science for women, University of Babylon, Babylon, 51002, Iraq
[4] Department of Cyber Security, College of Sciences, Al-Mustaqbal university, 51001, Babylon, Iraq
[5] Department of Computer Science, College of science for women, University of Babylon, Babylon, 51002, Iraq
Emails: zainab.abdulhameed@uokerbala.edu.iq, nihad.ab@uowa.edu.iq, ahmed.fanfakh@uobabylon.edu.iq, ali.kadhum.mohammed@uomus.edu.iq, esras.hadi@uobabylon.edu.iq

Abstract

Dynamic voltage and frequency scaling (DVFS) is a tool used primarily to decrease computer processor energy consumption by lowering its operational frequency. Their only downside is that they distract from the efficiency of parallel applications while operating on parallel platforms. In a heterogeneous cluster architecture, however, a genetic algorithm is being implemented and applied to model the best trade-off between energy-saving and parallel application performance degradation. The proposed algorithm selects the best frequency vector in order to accomplish these objectives by providing the same compromise. So, the objective function of the genetic algorithm at the same time gives limited energy consumption and minimum decreases in performance. The SimGrid simulator will be used for all experiments. The suggested algorithm saves the average energy by (20 %) and the application performance degrades to the limit (0.15 %).

Received: October 19, 2023 Revised: February 25, 2024 Accepted: June 19, 2024

Keywords: Genetic algorithm; heterogeneous cluster; frequency scaling; energy consumption

1. Introduction

In the past few years, scalable applications have grown dramatically for different scientific sectors with considerable attention has been paid to the energy consumption problem in parallel and distributed systems. System requirements have forced system engineers to build increasingly efficient architectures to deal with insolvent or highly complex scientific issues, which require huge resources, including memory and computer performance [1]. As a result, the researchers have discussed many optimization challenges in a heterogeneous parallel computing system such as cloud or grid, to optimize the speed where parallel applications are being implemented. The necessity for design successful parallel programming has increased the need for more computational power. High computing power is obtained

by speed-up task execution by increasing the number of processors to form high-performance computing systems (HPC) or increasing the processor frequency until the thermal threshold is achieved. HPC nodes are increased to improve the computing performance for systems and apps. Therefore, large quantities of data that require strict performance can be processed, such as digital signal processing, data storage, and image processing, etc. [2]. The paper has the following structures: Part (2) introduces the relevant documents and part (3) provides a proposed method, Section (4) shows the training and testing stages and the final section shows the experimental results obtained by the method proposed.

2. Related Work

This section presents some of the important work in the field of optimizing energy consumption.
In 2018, M. Yadav and K. Khanna [3], the need for exascale efficiency has led to make the modern computer systems working with upper limit operating frequency and bandwidth, resulting in prohibitive energy consumption and failure rates. To minimize this issue, this paper proposes an energy-saving strategy using the DVFS capability of Intel processors to reduce the energy consumption and power of processors. The strategy utilized offline profiling to assess frequency levels and the results showed that energy saving of up to 7% could be achieved on a quad-core Intel computer with four selected NAS NPB benchmarks.
In 2019, S. Imamura et.al. [4], the system of DVFS technology is used to reduce the CPU cores' frequency when the target storage unit is completely utilized in the case of I/O intensive operations. The link between I/O latency and core frequency was first examined in low latency SSDs and showed that the core frequency should be maximized for I/O latency reduction. However, it was found that during I/O waiting times in this case the CPU consumes high power wastefully. The evaluation indicates that the CPU power consumption is decreased by 41.4% and performance degradation is just 0.8%.
In 2019, E. André et.al. [5] introduce a daemon mechanism named Dynamic Uncore Frequency (DUF), which dynamically adapts the uncore frequency to reduce applications' energy consumption while restricting the performance degradation to a user-defined maximum. The evaluation indicates, while preserving the slowdown limit, that DUF substantially reduces power and energy consumption. This also demonstrates how uncore frequency can be used as a tool for performance enhancement. The evaluation of DUF shows that with less than 3.5% performance degradation, DUF can reduce the socket power consumption by more than 12%.
In 2019, V. Sundriyal et.al. [6], many energy-saving techniques use DVFS in a given multi-core processor without taking into account its applicability level. The paper shows that ignoring DVFS granularity as a design constraint while designing strategies for energy-saving will reduce the effectiveness of the strategy for certain applications categories. In particular, for three different Intel processor microarchitectures, DVFS applicability rates (called granularities) are calculated. If the DVFS granularity is ignored, the loss of energy-savings is also evaluated. Experiments show that the overall energy consumption could be increased by as much as 19% percent if granularity is not aware of energy-saving techniques.

3. Proposed Method

The proposed method used genetic algorithm to choose the optimal frequency vector, which achieves low energy consumption and minimal performance degradation simultaneously for a parallel iterative program as seen in the flowchart presents in Figure (1).

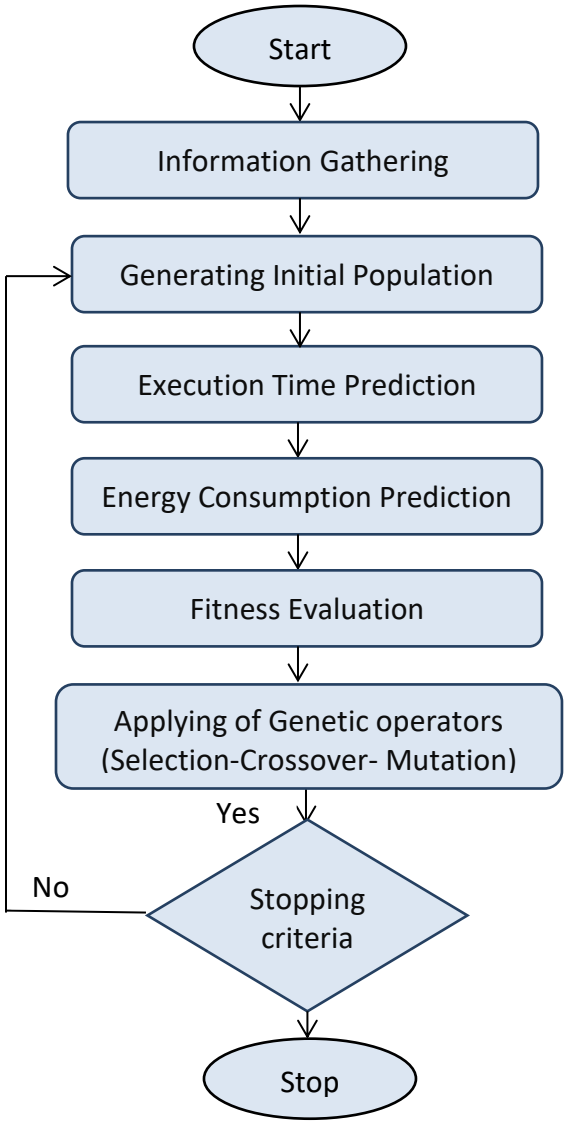

Figure 1. The flowchart for the proposed system

The details of the proposed method are provided as follows:

- Information Gathering: After compiling the parallel iterative program, the proposed method starts collecting data form it. The collected data are time of communication, computation time, static power, dynamic power, and frequency for each node in the heterogeneous cluster.

- Generating Initial Population: Based on the previously gathered data, the initial population consisting of a number of individuals will be generated. Each individual will represent a node's frequency. Each node operates at a specific frequency (genes) that is randomly selected within a certain range and according to the type of that node $Fi : j = 1, 2, ..., mi$, where mi is the number of available frequency of node i. Each group represents a candidate solution in the search area.

- Prediction of Energy Consumption and Execution Time: Models of energy consumption and execution time described in [7] are used for the evaluation of the generated cluster. The algorithm that predicts the energy consumption and the execution time of the iterative program is based on information gathered and used by the models.

- Fitness Evaluation: In fact, the dynamic power is reduced when the frequency of a processor is scaled down, but its processing power decreases proportionally. So, the execution time may be substantially increased. However, during

this period dynamic and static powers are both consumed. Therefore, the total energy consumption of the application cannot be optimal. The ratio between computation and communication time is different in heterogeneous parallel architecture and the static and dynamic power are also different. Thus, it is not easy for each processor to choose the best frequency [8]. The vector of scaling factors will be chosen according to the best balancing between performance and energy consumption. The aim of this paper is to reduce energy consumption and to prevent major increases in operating time. Therefore, as defined in algorithm 1, in a heterogeneous cluster by the genetic algorithm, the best frequencies will be chosen.

- **Applying of Genetic Operators**: To form the next population, genetic operators like selection, crossover, and mutation shall be used on population solutions generated.
- **Selection Method**: Tournament selection is one phase of a genetic algorithm that executing by select a random number of chromosomes (clusters) and choose one with maximum fitness than other chromosomes in the tournament.
- **Crossover Method**: The next important stage of the genetic algorithm is the crossover, using the (Uniform Crossover) that depending on a certain probability this process is performed, this probability value is referred to as PC.
- **Mutation Method**: The mutation process will be implemented by randomly changing the gene position of a specific individual with the gene positioning of another from the same type of node present in the predefined frequency vector. Also, depending on another probability value called PM.

The following genetic algorithm defines the vector for frequency scaling factors to provide a good possible compromise between a decrease of energy consumption as well as an execution time when a parallel program executing on the heterogeneous cluster.

Algorithm 1: Training Algorithm

Input:

T_{cp_i} : *array for all computation times for all nodes with the highest frequency after the execution is complete.*

T_{cm_i} : *array for all communication times for all nodes with the highest frequency after the execution is complete.*

Pop-size :population size.

Cluster-size: number of nodes maximum.

Max-gen: number of max generation.

P_{d_i}: *array of a dynamic power of all nodes.*

P_{s_i}: *array of a static power of all nodes.*

F_{max_i}: *array of a maximum frequency for all nodes.*

F_{min_i}: *array of a minimum frequency for all nodes.*

Output: *The Best optimization Vector of Frequencies*

1: $S_{cp_i} = \frac{max_{i=1,2,\dots,N}(T_{cp_i})}{T_{cp_i}}$

2: $F_i = \frac{F_{max_i}}{S_{cp_i}}$

3: Round each initial calculated Fi frequency in the closest available node

4: Generate the first population by choosing random frequencies in the range Fi to F_{min_i} *for each solution.*

5: $T_{Old} = max_{i=1,2,\dots,N}(T_{cp_i} + T_{cm_i})$

6: $E_{original} = \sum_{i=1}^{N}(P_{d_i} . T_{cp_i}) + \sum_{i=1}^{N}(P_{s_i} . T_{Old})$

7: Gen=0

8: ***Repeat***

9: *Gen = Gen+ 1*

10: ***for*** *i=1 to pop-size do*

11: ***for*** *j= 1 to cluster-size do*

12: $S_i = \frac{F_{max_i}}{F_i}, \ i = 1,2,\dots,N.$

13: $T_{New} = max_{i=1,2,\dots,N}(T_{cp_i} . S_i) + MinT_{cm}$

14: $E_{Reduced} = \sum_{i=1}^{N}(S_i^{-2} . P_{d_i} . T_{cp_i} + P_{s_i} . T_{New})$

15: $P_{norm} = \frac{T_{orginal}}{T_{predicted}}$

16: $E_{norm} = \frac{E_{predicted}}{E_{orginal}}$
17: $Fitness_i = P_{Norm} - E_{Norm}$
18: ***end for***
19: ***end for***
*20: Choose the individual with the method of selecting tournament ……. **see algorithm 2***
*21: The crossover method applying to the chosen parents…**see algorithm 3***
*22: The mutation method applying to the new parents...**see algorithm 4***
23: Applying steps 10-17 to evaluation the new offspring.
*24: Exchange the best new-generation individual with the worst randomly selected individual ……. **see algorithm 5***
*25: **until** Gen > = Max-gen **or** for a certain number of generations, the best individual doesn't change.*

Algorithm 2: Selection Algorithm

Input:
Population
S: five individuals from population
Output: Two *parent*
1: ***for*** *i= 1 to 2 do*
2: ***for*** *j= 1 to s do*
3: *parent = maximum fitness value of individual*
4: *return parent*
5: ***end for***
6: ***end for***

Algorithm 3: Crossover Algorithm

Input:
Two parents
f: 0 or 1
flip: true or false
M: number of nodes
Output: *Two parent after crossover*
1: ***if*** *flip=true* ***then***
2: ***for*** *i= 1 to m do*
3: ***if*** *f=1* ***then***
4: *change gen of parent with identical gen from the other parent*
5: ***end for***
6: ***else end algorithm***

Algorithm 4: Mutation Algorithm

Input:
Two parents
flip: true or false
Output: *Two parent after mutation*
1: ***if*** *flip=true* ***then***
2: ***for*** *1=1 to 2 do*
3: *check the type of node*
4: *r= select random frequency from the available node's frequencies*
5: *change random gen from parent instead of r*
6: ***end for***
7: ***else end algorithm***

Algorithm 5: Replacement Algorithm

Input:
Population
S: five individuals from population
P: parent with maximum fitness value
Output: *Population after replacement*
1: ***for*** *j= 1 to s do*
2: a= select the individual with smallest fitness value
3: ***if*** *a.fitness < P.fitness* ***then***
4: replacement P instead of a in population
5: ***else end algorithm***

4. Training and Testing Stages:

The best frequency vector which gives a good trade-off between the energy consumption and the performance of the parallel application is selected from many scenarios. The best sequence vector in the training mode is chosen by applying the offline genetic algorithm. Whereas, every new case for a program concerning new problem size in the testing mode is matched to the near case in the training table. The granularity of a parallel application is the ratio between the computations to communications as follows:

$$G = \sum_{i=1}^{n} T_{cp_i} \Big/ \sum_{i=1}^{n} T_{cm_i} \quad \dots (1)$$

Where Tcp is the time of the computations and Tcm the time of communications. Granularity value is used as a matching value in the testing mode.
However, the training table contains a set of records each with two fields, the frequency vector, and its granularity value.
In the testing mode, an online DVFS algorithm is proposed that works during the running of the iterative application. It gathers the computation and communication times after the first iteration of the iterative program. The granularity of these times is computed and matched with the training table granularity values. The frequency vector of the nearest match is selected and applied to all nodes of the cluster to save energy consumption. In the following, in the master node, the online DVFS algorithm works to choose the optimal frequency vector for cluster nodes.

Algorithm 6: Online DVFS

Input:
$\boldsymbol{T_{cp_i}}$: *array for all computation times for all nodes with the highest frequency after the execution is complete.*
$\boldsymbol{T_{cm_i}}$: *array for all communication times for all nodes with the highest frequency after the execution is complete.*
Output: *best energy consumption ratio*
1: ***if*** *the computing node is the master node,* ***then***
2: Gathering all computation and communication times in the vectors T_{cp_i} *and* T_{cm_i}
3: Compute the granularity value $\boldsymbol{G\_test} = \frac{\sum_{N}^{i=1} Tcpi}{\sum_{N}^{i=1} Tcmi}$
4: Compute the absolute error vector between G-test and G-train:
$\boldsymbol{Error_i = |G\_test - G\_train|}$
5: Select the frequency vector of the smaller error value of vector $error_i$ *in the training table.*
6: Apply the new frequency to the master node.
7: Send the frequency value for each node in the cluster.
8: ***endif***
9: ***if*** *not master node,* ***then***
10: Receive the new frequency from the master node
11: Apply the new frequency to the node's processor.
12: ***endif***

5. Experiments

5.1 Experiments Configuration

To validate the proposed method, this work is implemented under the Linux operating system. The SimGrid / SMPI simulator uses to simulate parallel cluster and executing message passing library MPI program. A simulation using SimGrid / SMPI simulator configured the heterogeneous cluster that running the parallel iterative program to apply the experiments. This simulator includes tools that are flexible to build the architecture of a cluster to execute messages passing programs on it [9]. The simulator has generated a heterogeneous cluster consisting of four main node types. Each type of node has characteristics that are different from the other nodes in terms of static and dynamic power, computing power (FLOPS), and the number of available frequencies for each nodes type. Details of these nodes are shown in table (1).

Table 1: Nodes Details

Node type	Max freq. GHz	Min Freq. GHz	Increment In GHz	Dynamic power	Static power
1	2.5	1.2	0.100	20 w	4 w
2	2.6	1.6	0.133	25 w	5 w
3	2.8	1.2	0.100	30 w	6 w
4	3.4	1.6	0.133	35 w	7 w

Whereas, the proposed method mainly depends on the energy consumption and performance prediction models. It is shows the verification of the accuracy of the two prediction models on the various heterogeneous nodes that are 4,8,16,32 and 64. The heterogeneous cluster comprises 4 types of nodes in which an equivalent node number for each type is used. A Parallel Message Passing Interface (MPI) uses Jacobi's method is applying on a parallel application. It is an iterative algorithm to evaluate solutions for the linear equation diagonally dominant systems as shown in (2), if the convergence criterion was satisfied, the iterative process is ended.

$$\begin{aligned} a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)} &= b_1 \\ a_{21}x_1^{(k)} + a_{22}x_2^{(k+1)} + \cdots + a_{2n}x_n^{(k)} &= b_2 \\ a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn}x_n^{(k+1)} &= b_n \end{aligned} \quad \text{.....(2)}$$

A genetic algorithm is a method of searching for approximate solutions which are very close to the exact solution. The suggested approach uses the genetic algorithm to find the best frequencies vector for each CPU in a computing cluster. GA manages a population of potential solutions where each chromosome represents a solution of frequencies vector for the problem. When this application is over, the genetic algorithm collects the data it needs. Table 1 explains the information gathered of each cluster node which is frequency, dynamic and static power, communication time, and computation time. The algorithm predicts the execution time and energy consumption of any vector based on all this information set of the scaling frequency factor. To select the best vector of frequency, several times the algorithm has been replicated. Thus, the accuracy of the prediction of the energy consumption model and execution time model is verified according to algorithm 1.

Tournament selection is one step of a genetic algorithm use for select a random number of chromosomes (clusters) and choose the individual with maximum fitness than other chromosomes in the tournament. The next important stage of the genetic algorithm is the crossover operator, where the uniform crossover uses a predefined probability value called PC probability. Also, another probability value called PM is used in mutation operation (Random Resetting) by selecting a random gene that is replaced with another from the same type of node present in the predefined frequency vector. The range of possibilities is set to 0<PM<1 and 0 <PC<1[10]. The nine PM and PC probability values used in experiments were shows in Table (2).

Table 2: Probability Value

PC	0.8	0.8	0.8	0.9	0.9	0.9	1	1	1
PM	0.3	0.2	0.1	0.3	0.2	0.1	0.3	0.2	0.1

Three population sizes are used in the experiments that are 100, 200, and 300. Four sizes of the problem have used for training (2000, 4000, 6000, and 8000). Therefore, there are 20 instances for each population size that is applied to all problem size. The results of the best scenario provide a balance between the highest rate of consumed energy saving with the lowest rate of performance degradation and will be presented in the next section.

5.2 Experiment results

The results were divided into two parts according to the stages obtained them, Training and Testing Result.

5.2.1 Training Result

The energy consumption model relies primarily on the execution time model, since the execution time correlated linearly to static energy and the computation time was correlated to the dynamic energy. However, the actual execution time was compared with a predicted execution one to verify the efficiency of the proposed algorithm in the training phase, as well as the real and predicted consumed energy are compared. The comparison indicated that the proposed model is accurate as the problem size of 6000 is applied to five nodes scenario as in Figure (1). The average error between the real execution time and the predicted one is about 6.07%.

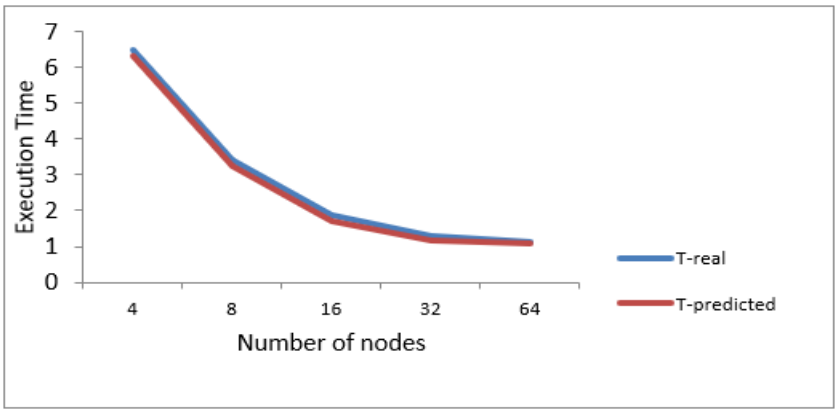

Figure 2. The execution time results

The total energy consumption was evaluated by the energy consumption model for each case with and without DVFS. Similarly, the execution time was also measured with and without applying DVFS. The energy-saving and performance degradation percentages were then determined for each situation. The experiments show that the proposed algorithm reduces energy consumption significantly.

The best scenarios are presented from many experiments results for values of performance degradation and energy saving. The ratio of the energy saving is 22.83% of the problem size 2000, 22.66% for problem size 4000, 27.34% for problem size 6000, and 24.86% of the problem size 8000. The algorithm, however, tries to minimize performance degradation.

In addition, these results are the values of average energy saving from several experiments with performance degradation. It also shows that the percentage of energy-saving decreases with increasing numbers of computing nodes.

Table 3: Best scenarios of several nodes

Problem-size	Scenario details			Number of nodes	Energy-saving	Perf. Degr.	scenarios name
	Pop-Size	PC	PM				
2000	100	1	0.1	4	21.77	-3.55	S1-2000
	200	1	0.1	8	22.13	-2.59	S2-2000

	300	0.9	0.1	8	22.83	-2.57	S3-2000
4000	100	0.9	0.1	4	21.74	-1.95	S1-4000
	200	0.8	0.2	16	22.66	-0.77	S2-4000
	300	0.9	0.3	16	21.98	-4.05	S3-4000
6000	100	0.9	0.3	16	24.36	-8.73	S1-6000
	200	0.9	0.1	32	27.34	-6.59	S2-6000
	300	1	0.3	16	23.08	-3.16	S3-6000
8000	100	0.9	0.2	4	22.89	-3.72	S1-8000
	200	0.8	0.1	32	24.86	-6.14	S2-8000
	300	0.8	0.1	32	24.95	-5.04	S3-8000

This degradation is because the communication times are increased as the program runs on a larger number of nodes. Through analysing the times of execution, results indicate that percentages of performance degradation in most cases decrease or increase performance when working with a large number of nodes. That is due using more nodes increase the communication time. Moreover, reduction of these nodes' frequencies will also decrease their efficiency.
The results also show that there are positive and negative values for the performance degradation ratio. The positive values are a degradation of performance, while the negative ones relate to improved performance relative to the cluster execution time without DVFS. The best scenarios chosen from all the results obtained in Table (3) are detailed. Details of the best scenarios for all problem sizes can be found in Figures 2,3,4 and 5.

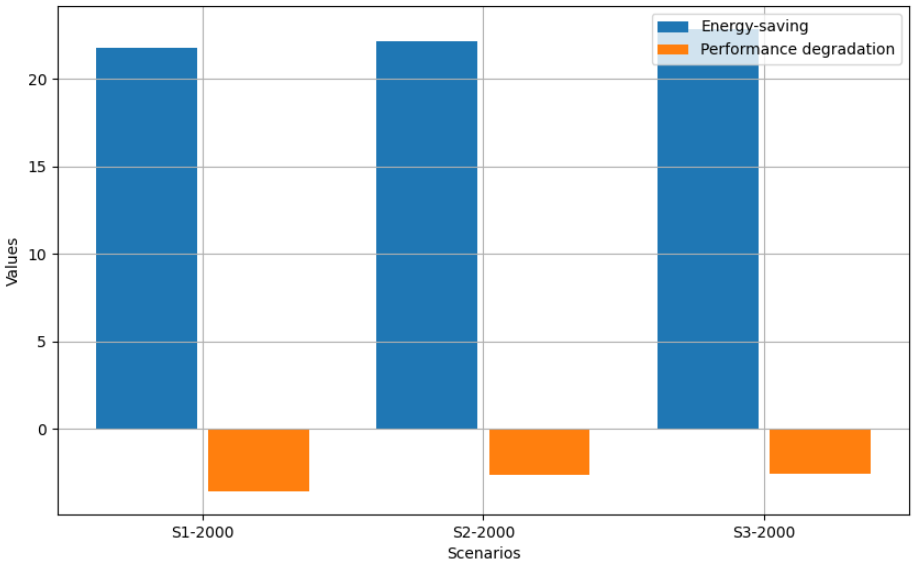

Figure 3. Best scenario for population size 2000

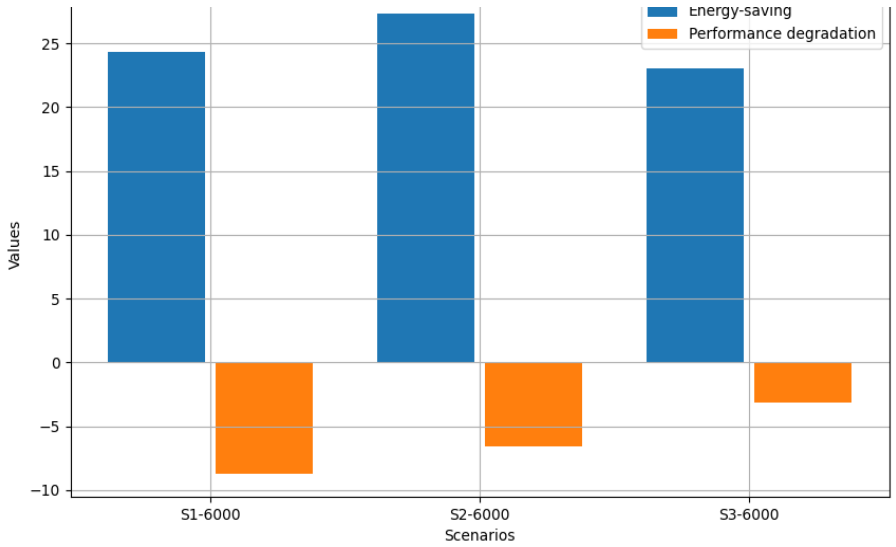

Figure 4. Best scenario for population size 6000

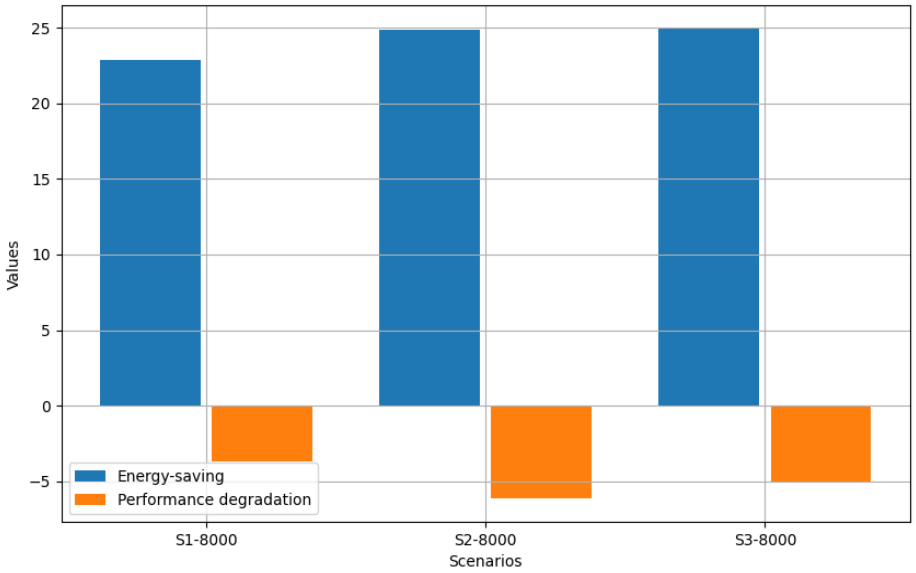

Figure 5. Best scenario for population size 8000

5.2.2 Testing Results

The online DVFS algorithm 6, is applied to a different instance of the parallel Jacobi method. Multiple scenarios have been used to test the ability of the proposed online DVFS algorithm. Four different sizes for the problem solves by the Jacobi method are selected that are 1000, 3000, 5000, and 7000. The results of the algorithm are demonstrated in the table (4). On average, the proposed model degrades performance by 0.15% while saves the energy consumed by 20%. The results show that the increase number of nodes leads to a decrease in the energy-saving ratio because this increase leads to a decrease in the computation that is linearly correlated with the dynamic energy consumption.

Table 4: Result of testing mode

Problem size	No. of Nodes	Energy-saving	Performance Degradation
1000	4	21.911473	-0.02976422
	8	23.486522	-1.51734937
	16	20.342238	3.8077763
	32	16.307242	1.35966426
	64	10.006609	-1.38603289
3000	4	21.099917	1.3504645
	8	20.156972	-4.61068239
	16	23.669503	1.69524059
	32	21.226512	-0.6603379
	64	15.031051	-1.98168801
5000	4	17.122028	3.74955382
	8	18.771995	2.75142555
	16	21.737419	-3.60722075
	32	15.40181	7.77029344
	64	18.838914	0.15064918
7000	4	21.343213	-1.38726849
	8	23.159847	-4.30149006
	16	23.151607	1.2897435
	32	25.416811	0.54984041
	64	20.899305	-1.90372027

6. Conclusion

This paper introduces a method for utilizing a genetic algorithm in an offline training mode to construct a testing table. The training outcomes are derived from executing a variety of scenarios, which include diverse genetic parameters, varying problem sizes addressed by the parallel application, and different quantities of computing nodes. In the subsequent testing mode, an online Dynamic Voltage and Frequency Scaling (DVFS) algorithm is applied to a distinct instance of the parallel Jacobi application. Experimental results indicate an average energy consumption reduction of up to 20%, with a performance degradation of merely 0.15%.

References

[1] J. L. Hennessy and D. A. Patterson, “Multiprocessors and thread-level parallelism,” Comput. Archit. A Quant. Approach, pp. 196–264, 2006.

[2] P. J. Denning and W. F. Tichy, “Highly parallel computation,” Science (80-.)., vol. 250, no. 4985, pp. 1217–1222, 1990.

[3] M. Yadav and K. Khanna, “Energy Saving Strategy Based on Profiling,” International Journal of Scientific Research in Science, Engineering and Technology (ijsrset.com), 2018.

[4] S. Imamura, E. Yoshida, and K. Oe, “Reducing CPU Power Consumption with Device Utilization-Aware DVFS for Low-Latency SSDs,” IEICE Trans. Inf. Syst., vol. 102, no. 9, pp. 1740–1749, 2019.

[5] E. André, R. Dulong, A. Guermouche, and F. Trahay, “DUF: Dynamic Uncore Frequency scaling to reduce power consumption,” 2019.

[6] V. Sundriyal, K. Keipert, M. Sosonkina, and M. S. Gordon, “Effect of frequency scaling granularity on energy-saving strategies,” Int. J. High Perform. Comput. Appl., vol. 33, no. 4, pp. 590–601, 2019.

[7] Z. A. Abdulazeez, A. B. M. Fanfakh, and E. H. Alwan, “Selecting Best CPU frequency for energy saving in cluster using genetic algorithm,” in IOP Conference Series: Materials Science and Engineering, 2020, vol. 928, no. 3, p. 32073.
[8] Jean-Claude Charr , Raphael Couturier, Ahmed Fanfakh and Arnaud Giersch, Consumption Reduction with DVFS for Message Passing Iterative Applications on Heterogeneous Architectures. The 16 th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing. pp. 922-931. IEEE Computer Society, INDIA (2015).
[9] A. Kumar, R. C. Gupta, and P. Sharma, "A Genetic Algorithm Approach for Energy Efficient CPU Frequency Scaling in Cluster Computing," Journal of Supercomputing, vol. 78, no. 1, pp. 123-140, Jan. 2022. DOI: 10.1007/s11227-021-03845-9.
[10] L. Zhang, Y. Chen, and D. Wang, "Dynamic Frequency Scaling for Energy-Aware Cluster Computing Using Genetic Algorithms," Future Generation Computer Systems, vol. 120, pp. 242-250, Mar. 2021. DOI: 10.1016/j.future.2021.01.007.
[11] Idrees S.K., Fanfakh A.B.M. (2018) Performance and Energy Consumption Prediction of Randomly Selected Nodes in Heterogeneous Cluster. In: Al-mamory S., Alwan J., Hussein A. (eds) New Trends in Information and Communications Technology Applications. NTICT 2018. Communications in Computer and Information Science, vol 938. Springer, Cham.
[12] Ali, M. Z., Awad, N. H., Suganthan, P. N., Shatnawi, A. M., & Reynolds, R. G. (2018). An improved class of real-coded Genetic Algorithms for numerical optimization.Neurocomputing, 275, 155–166. doi:10.1016/j.neucom.2017.