Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 6, July, 2021: 8816 - 8829

Research Article

Energy Aware Scheduling of Tasks in Cloud environment

Dhirendra Kumar Shukla[1], Shabir Ali[2], Munesh C. Trivedi[3]
[1,2] CEA Department, GLA University, Mathura, 281406 (India)
[3]CSE Department, NIT, Agartala, 799046 (India)
[1]dhirendra.shukla@gla.ac.in, [2]shabir.ali@gla.ac.in, [3]munesh.trivedi@gmail.com

Abstract: Energy consumption in cloud data centers is increasing as the use of such services increases. It is necessary to propose new ways of reducing energy consumption. Task scheduling is an essential aspect of energy consumption in a cloud data center. The main objective of this research is to efficiently schedule various workflows so that energy consumption is minimized in the overall makespan and slice time of the processor. In this paper, the author makes a new approximation for the dynamic allocation of workflow tasks to currently available resources using genetic algorithm crossover, mutations, and evaluation operators. The purpose of this scheduling is to reduce the makespan and slack time on the processor and thus reduce the energy consumption of the processor in times that are not in use.

Keyword: - Makespan, Workflow Scheduling, Energy consumption

I. INTRODUCTION

Tasks from the clients/users to be submitted to the cloud for processing requires some processing capacity, network bandwidth, has some deadline, and to be completed in minimum time. The services provided by the datacenters to these application tasks are from the datacenter components housed within it. These tasks are thus to be scheduled in the datacenter and thus to be ready for execution within some of the dedicated servers/processors within the datacenter as per the requirements of the applications. In fact, service providers have to cope with Scheduling of the tasks, reducing makespan and reducing energy consumption. Makespan refers to the time of completion of the last task among the number of tasks scheduled for processing. The field of cloud computing which deals with the reduction in energy consumption is known as green computing because an enormous increase in the consumption of energy consumption leads to the increase in the emission of carbon dioxide from these datacenters which in turn effects the environment. So to make a cloud computing efficient and thus ecofriendly a fair and optimal scheduling of tasks is thus an important factor so that tasks can be completed in minimum span of time and thus energy consumption can be reduced by applying some of the techniques for reduction of energy consumption.

II. RELATED WORKS

To reduce the power consumption and makeup of computation processing, a lot of work has been done in scheduling data centers. A lot of algorithms have been proposed for the scheduling of tasks from certain application in cloud environment. Some of the algorithms deal with the scheduling of tasks

intending to reduce the cost of execution, minimizes the makespan, reduce energy consumption, and meet the application's deadline, thus completing the requirements (cost, makespan, energy consumption, etc.). There were extensive reviews of workflow scheduling in distributed systems. Since this puzzle is NP-hard, heuristics are used by most current workflow scheduling strategies to include uncertainties. Typically, the primary part of workflow scheduling techniques such as HEFT [1] consolidates an objective. Shukla.et.al [2] proposed multi-objective algorithms to reduces makes pan and energy consumption in cloud environment. The main focus is on metaheuristic techniques such as genetic algorithms (GA), ant colony optimization (ACO), and particle swarm optimization (PSO). Mezmaz et al.[3] reviewed high-performance computing systems, especially pre-compressed parallel applications. In this, parallel applications have been designed to overcome completion time without much focus on energy consumption. Ying et al. have introduced two energy-conscious GA task scheduling algorithms that analyze makespan and energy consumption [4]. As a two-objective optimization strategy, there is an acceptable agreement between makespan and energy consumption. This method is based on DVS to reduce energy consumption. To define fitness, they use combined and dual fitness methods and select experiments from individuals that suggest that both of these algorithms can optimize makeup and energy more efficiently. Tao et al. have proposed a more comprehensive and accurate model for OSCR (optimal timing of computing resources) [5]. The model was designed for heterogeneous resources in the cloud world, MacPassan, and optimization is considered energy consumption from both economic and ecological perspectives. For this multi-purpose optimization puzzle, the design must obtain the Pareto solution. The multi-parent crossover operator (MPCO) design and two-step algorithm problem is based on the standard multi-objective genetic algorithm of the Pareto-solution-based hybrid genetic algorithm (CLPS-GA) in state archives. The simulation result is introduced to determine and establish the effectiveness of CLPS-GA in terms of convergence, stability, and solution diversity. Yes. Babukartik et al. offered a hybrid algorithm [6] that combines the benefits of ACO (ant colony optimization) and cuckoo search. Functions were determined based on reducing the energy consumed. Experimental findings suggest that the efficiency of the hybrid algorithm is much improved. The author [7] has proposed improving the throughput, stability, and lifetime of sensor networks based on cluster networks. Xia Zhu et al. suggest a rolling-horizon scheduling design and energy consumption model [8]. Shukla.et.al [9,10] have proposed task scheduling algorithms in heterogeneous and homogeneous environment to minimizes makespan and resource utilization. For standard, independent specific functions, they introduced EARH with a novel energy-aware scheduling algorithm. To improve scheduling efficiency, the EARTH system employs a rolling-horizon optimization technique. In addition, techniques for resource scaling and resource scaling are generated and incorporated into EARH, which can flexibly change the size of the active host to meet the real-time requirements of the task and save resources. Young et.al [11] have on such systems with apparently heterogeneous resources, we deal with the planning problem for pre-determined parallel applications, which are responsible for both the absolute time of application, namely, maple and energy consumption. In order to reduce energy consumption, the scheduling algorithm adopts dynamic voltage scaling (DVS) [12], with the integration of RS and MCER significantly contributing to reducing energy consumption. Using the DVS (Dynamic Voltage Scaling) technique, a recent advance in processor design, the energy-saving of ECS (Energy-Aware Scheduling) is allowed. Offers positive outcomes that highlight the value and ability of DVS in reducing energy consumption. Some of the previous cloud work for planning application activities considering a single goal or multi-goal for optimization (cost, makespan, and energy), etc [13]. Some research papers focusing on various scheduling techniques in cloud environments were considered and summarized [14]. The scheduling algorithm used

to reduce the makespan of workflow applications is presented in which represents a genetic algorithm-based scheduler to reduce the makepan of cloud [15,16]. The authors [17] the focus is on reducing the makespan of heterogeneous climate. It has been suggested that [18] and [19] extended scheduling algorithms and maximum-minimum strategy Meta heuristic algorithms to reduce workflow applications based on anti-colony optimization. In the multi criteria algorithm, we investigated those that optimize makespan and energy consumption. By first optimizing a criterion, i.e. makepan, they use a two-step optimization process and then reschedule to find a solution that minimizes energy consumption [20,21].

An application is a workflow with multiple functions that are associated with preceding constraints. Each task will be executed, and an output dataset will be provided. This dataset is then sent to the next task specified by the workflow structure. The workflow typically consists of a DAG structure (cyclic graph directed): a graph containing function nodes, and preceding constraints. An objective DAG graph is denoted as G = (V, E), where V is a function and E as a dependency between relations and a priority range, and E V is the edges or functions of a node. in the middle. Each task in this graph has a weight w (vi), which is the length or equal number of instructions of the task, and the load w adds the data transfer rate between the jobs (ei, j) of the edges. However, a communication cost is required only when the two tasks are assigned to different processors. In other words, when assigning tasks to the same processor, the communication cost is zero and thus can be avoided. The weight marked as w (vi) on the vi function indicates the expense of calculating the function. The computation time of work on a Pj processor is also expressed as w (ei, j). The average estimated cost of a job in a directed cyclic graph is represented by the average cost of contact between two nodes. Advanced programs are compute-intensive, and all are interaction-intensive. The communication to computation ratio (CCR) is a metric that indicates a communication-intensive, computationally-intensive or [22] output graph. The average communication cost of the CCR factor is divided by the average cost calculated on a target rule.

III. PROPOSED APPROACH

Our proposed approach efficiently allocates the makespan and processor to overcome energy consumption and use resources adequately. The problem is that workflow schedules are used to schedule between possible sets of schedules in multiprocessor environments, thus retaining the preceding relationship between maple and tasks to reduce energy consumption. The method implemented for energy-efficient scheduling in data centers trades with various workloads and considers multiple varieties in the application by estimating different workloads. In this paper, an efficient task scheduling method is proposed to deal with makespan removal. Energy consumption in the data center will be minimized, thus developing a green computing environment. Existing techniques HEFT [1], FCFS, MAX-MIN, and MIN-MIN have been compared and implemented with the proposed methodology. Resource allocation and resource determination are essential aspects that affect networking, parallel, distributed computing and cloud computing. Many researchers have suggested efficient allocation of various algorithms so that they can plan cloud resources. In three steps, the cloud scheduling process can be simplified, such as searching and filtering resources. The datacenter broker locates the resources present in the network system and collects the status information related to them. Selection of resources: The target resource is selected based on certain goals and resource criteria. The defining step is this. Submitting a task - This task is submitted for the selected resource.

A. **PROPOSED ENCODING:** Each solution is encoded as a chromosome in a genetic algorithm (GA) approach [23]. The length of a chromosome is the N gene for each chromosome. In workflow scheduling, each schedule is in chromosome form. The schedule contains the purposes of application as a vector with late blocking. First, the functions of the graph need to manage the preceding relationships based on their effect on other tasks in the execution graph. Second, tasks require a map to the appropriate resources of the collection of potential resources. Each chromosome is a sequence of numbers in a permutation encoding, representing a number in the order or order of execution of tasks from an application.

a) **INDIVIDUAL REPRESENTATION:** Another execution preference of dependent tasks within a workflow can significantly affect the performance of workflow execution. Accordingly, it was also claimed to provide a task allocation scheme for each resource in the encoded operation chain [24] [25]. The main purpose for the scheduling-order string is to code the task dependency and the execution dependency for the tasks assigned to the associated priorities / processors. In this proposed work, a workflow scheduling problem is a possible solution that needs to satisfy the following constraints:

- A task can be ready to start only after the completion of all its predecessors.
- Every task comes once and only once in the schedule.
- Every task must be allotted to the ready time slot of the enabled resource/processor to work.

Each person in the search space represents a workable solution to the issue and enters the function vector without violating the past relationship. To encode the service allocation for each task, the workflow scheduling problem involves operation strings. The order of execution of interdependent tasks in a workflow is controlled by their dependencies, such that a task is always executed after the tasks of its parent. However, on the same processor / resource, multiple independent tasks will compete for the same time slot in a workflow.

B. **INITIAL POPULATION**- It is assumed that a collection of multiple possible solutions (chromosomes) is named a population. In a standard genetic algorithm, the initial population is generated randomly. We used t-level and b-level [26] tasks in a workflow to pick the individual population for our proposed work. The issue here is to create a good and main objective initial population that would lead to rapidly knowing the solution. We have considered the initial population consists of two individuals for making the initial population for this goal. Each hires individuals in the execution sequence of an app's tasks. The first relates to the execution sequence of the tasks arranged from the entry task according to the jobs' decreasing b-level. The second one corresponds to the tasks' execution sequence according to the increasing t-level of the tasks, starting from the entry task.

a) **BOTTOM LEVEL (B-LEVEL) CALCULATION:** B-level functions are defined by DAG (Directed Cyclic Graph), which starts with the exit function of DAG. The B-level value for the exit process is zero, that is, the B-level (exit) = 0.0. Using floor level as a metric of priority, incoming correspondence to the node is not taken into account. In the target scheme, all preceding nodes except ancestor nodes have not been previously determined. Although the two nodes are in the same stage, there can be a significant difference in their incoming communication costs. For a wider resource choice, the person with more important

communication should be marked first. By including incoming contact costs, the lower level can be "increased in the top direction".

$$b - level(v_i) = \overline{w_i} + \max_{v_j \in succ(v_i)}(\overline{c_{i,j}} + b - level(v_j)) \text{ .....(1)}$$

In equation (1), succ(v_i) belongs all the successors of node v_i and $\overline{c_{i,j}}$ belongs to the average communication cost from node v_i to node v_j.

b) **TOP LEVEL (T-LEVEL) CALCULATION:** The T-level [39] calculated the functions of the directed acyclic graph (DAG) based on the input function of the directed acyclic graph function (DAG). For an entry function, the t-level value is zero, that is, the t-level (ventry) = 0.0. The node's T-level varies during the scheduling process, while the B-level is usually constant until the node is determined. The T-level varies because when two incident nodes are constructed from the same processor, the weight of one edge can be zero. The path reaches a node whose length determines the node's T-level. Thus, the longest can appear. The t-level for the successive nodes of an entry task for graph is thus calculated as:

$$t - level(v_i) = \max_{v_j \in pred(v_i)}(\overline{c_{j,i}} + t - level(v_j) + \overline{w_i}) \text{ .....(2)}$$

In equation (2), pred(v_i) belongs all the predecessors of node v_i and $\overline{c_{j,i}}$ belongs to the average communication cost from node v_j to node v_i.

Table 1. Communication cost matrix for graph as shown

Task	**Node**													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	**8**	**18**	**36**	**1**	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	**8**	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	18	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	**36**	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
5	0	0	0	0	0	0	**23**	**5**	**28**	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	**23**	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	5	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	**28**	0	0
9	0	0	0	0	0	0	0	0	0	0	**28**	20	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	**28**	**20**
11	0	0	0	0	0	0	0	0	0	0	0	0	0	**40**
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P1	P2	P3	P4	P5
0	13.44	16.13	8.96	10.08	20.16
1	2.88	3.45	1.92	2.16	4.32
2	6.28	7.53	4.18	4.71	9.42
3	12.11	14.53	8.07	9.08	18.16
4	0.55	0.61	0.33	0.38	0.76
5	5.02	6.02	3.34	3.76	7.53
6	7.81	9.38	5.21	5.86	11.72
7	1.72	2.07	1.15	1.29	2.58
8	9.55	11.46	6.36	7.16	14.32
9	3.38	4.06	2.25	2.54	5.08
10	9.34	11.21	6.23	7.01	14.02
11	6.82	8.18	4.54	5.11	10.23
12	10.98	13.18	7.32	8.24	16.48
13	13.49	16.18	8.99	10.11	20.23

Nodes	B-Level	T-Level
0	225.93	0.00
1	204.17	4.06
2	189.17	8.92
3	157.77	17.39
4	80.80	0.66
5	193.22	10.31
6	165.08	21.50
7	109.38	10.31
8	78.56	23.97
9	134.08	28.16
10	102.61	41.62
11	40.78	37.92
12	65.05	56.75
13	10.11	76.11

According to the decreasing b-level value from the starting task to the ending task of the directed acyclic graph, the order of execution of tasks is i.e. S1 : 0,1,5,2,6,3,9,7,10,4,8,12,11,13. And the order of performance of the tasks for the same graph according to the increasing t-level starting from the entry task is i.e. S2 : 0,4,1,2,5,3,6,8,7,9,11,10,12,13.Thus, for the first generation of our approach, there are two individuals S1 and S2 that have similar fitness values, namely the make-up for execution in available processors 74.70 and 81.29, respectively.

C. CROSSOVER: An integral aspect of the genetic algorithm is crossover [27] and mutation. Stability is mainly affected by both of these operators. Somehow, the chromosome has to be conscious of a solution that represents it. A chain of genes constitutes each chromosome. We have assumed the timeliness order of the function in DAG scheduling, which holds the primary constraint as a chromosome, and each role in this sequence is considered a gene. There are many other encoding approaches. This is mainly dependent on the problem that was solved. For example, one can directly encode integer or real numbers; often, encoding such permutations is beneficial. In ordering problems, permutation encoding can be used. After deciding which encoding, we will use, the paper will take another step toward crossover. The crossover selects genes from the parent's chromosomes and creates a new progeny and, namely, the relation of precedence must also be preserved by a schedule. The best way to do this is to randomly select such crossover points and copy everything from the first parent to the copy after switching the crossover point from the second parent. Thus, the latest progeny maintains a connection between the functions of an application. The crossover operation based on the permutation encoding method for the two schedules S1 and S2 is shown in Figure 1 as individual 1 and individual 2, as related above.

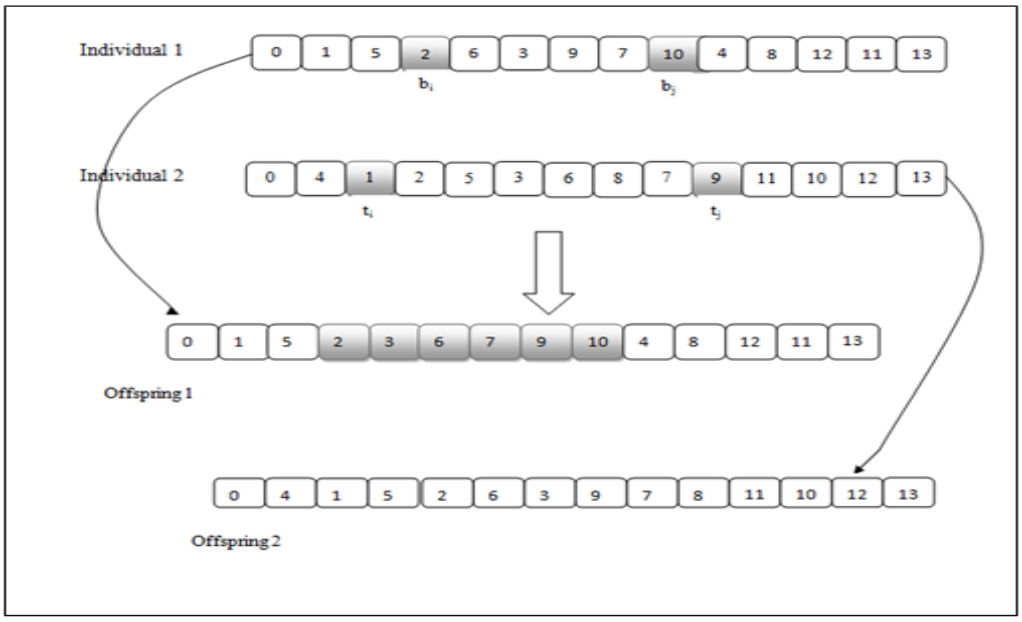


Figure 1. A typical representation of permutation encoding based Crossover operation

D. MUTATION: -Mutations occur after crossover [27]. This is to avoid the problem solved by sliding into the local optimum of all solutions in the population. New progeny is added at random by mutation. Figure 10 shows a standard representation of the permutation-based encoding mutation process. Mutations depend on encoding as well as crossover. For example, when we are doing permutations, mutations can exchange two genes. Nevertheless, thus the scheduling order obtained after the exchange of genes (functions) must maintain the same preceding relationship as the application graph for the task.

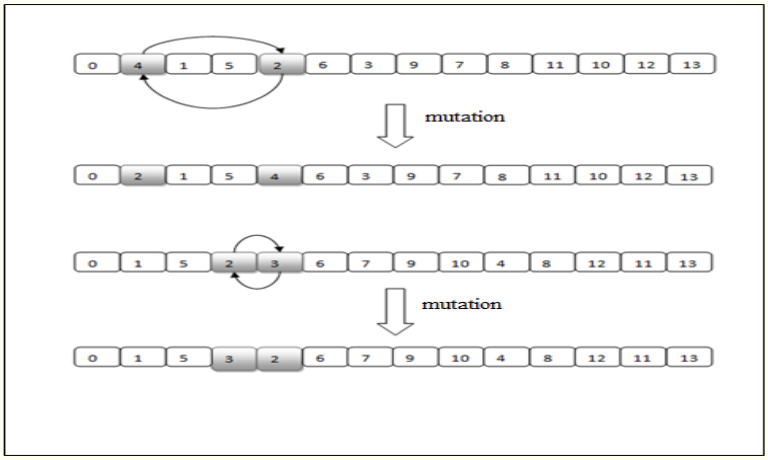


Figure 2. A typical representation of permutation-based encoding Mutation operation

Two basic GA parameters are available [28] - crossover probability and mutation probability. Crossover probability refers to how well the crossover has performed. There are no crossovers due to parents having close copies of the offspring. If there is a crossover, offspring are formed from parts of the parent's

chromosome. If the probability of crossover is 100%, then all offspring are created by crossover. If it is 0 percent, then the whole new generation is made up of duplicate copies of the chromosomes of the old population. The crossover has been established in the hope that the new chromosome will contain good parts of the old chromosome and perhaps the new chromosome will be stronger. But to survive, it is good to leave some part of the population to the next generation.

IV. WORKFLOW OF THE PROPOSED METHOD

A. **Energy Consumption:** In this work, the Dynamic Voltage Frequency Scaling (DVFS) [11] system has been used to reduce energy consumption; each resource will operate at various voltage and frequency ranges. Let M has considered as a set of nodes $\{N_1, N_2, N_3, \ldots, N_n\}$ that can work in a voltage limit of N_{mini} and $N_{max\ i}$, T is a set of tasks {T1, T2, T3, …, Tj} to be performed using N and ST has a bearing of subtasks {ST1, ST2, ST3, …, STs} for each task Tj. If the expected performance frequency of a subtask has taken as fST, then the machine's power consumption for operating the subtask can be estimated using equation (1,2,3).

$$\text{Energy}_{\text{buzy}} = \sum_{i=1}^{n}\sum_{j=1}^{k} \lambda \times v_j^2 \times f_j \times F(i,j) \times T(i,j) \qquad (1)$$

$$\text{Energy}_{\text{Idle}} = \sum_{j=1}^{k} \lambda \times v_{\text{lowest}}^2 \times I_j \times f_{\text{lowest}} \qquad (2)$$

$$\text{Energy}_{\text{total}} = \text{Energy}_{\text{buzy}} + \text{Energy}_{\text{idel}} \qquad (3)$$

Where, E_t can also be illustrated as

$$\text{Energy} = \text{Energy}_{\text{buzy}} + \text{Energy}_{\text{idel}}$$

Where, λ has a switching is associated with a constant action factor and can be achieved by doubling the flip frequency (number of switches per clock) with load capacitance, for this function the required resource for voltage is i which reduces work j is. And f work. Is the frequency of doing λ is the constant parameter related to the dynamic power, vj is the supply voltage at which the processor j is regulated, f is the voltage frequency In practical scenarios it is possible that a processor j does not operate at all times and remains idle for some period i (j). In this way it is necessary to keep an account of the energy idle as well. It is assumed that during the idle cycle the processor operates at the lowest possible voltage current and the lowest possible frequency current. And total energy consumption is the sum of the energy consumed in total busy cycles and the energy consumed in idle cycles.

The sequence of tasks to be performed in the primary tasks is constrained to find the minimum makespan. Thus, the minimum slack time so that the system consumes less energy upon hibernation is shown in the flowchart, as shown in Figure 11. The order of operations performed in a directed acyclic graph (DAG), which shows dependencies between predefined functions, is under

1. Find the initial population i.e. the schedules based on t-level and b-level of tasks.
2. Evaluate the fitness value i.e. makespan for the individuals (initial population schedules).
3. Perform crossover and mutation on the individuals up to the population size to make new individuals for the same generation.

4. If population size reaches the maximum population forwards the fittest individuals to the next generation.
5. Performs the energy computation and makespan of the generation.
6. Perform the same operation for each generation until it reaches the maximum number of generations to find the best schedule and thus the minimum makespan.

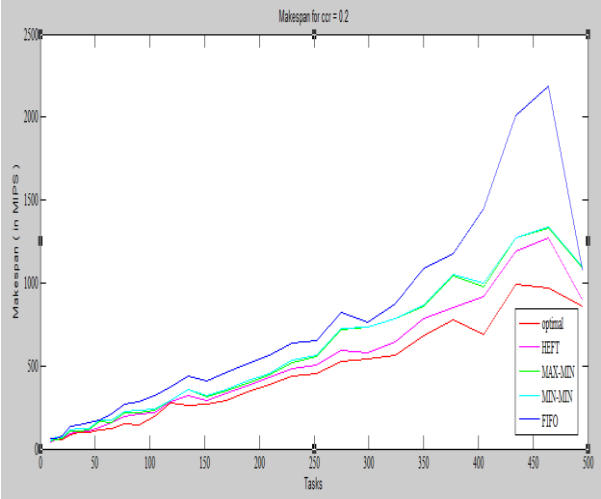
V. EXPERIMENTAL RESULTS

Simulation results for the research work are presented under the following parameters. The number range of functions for the communication of variable cost ratio (CCR) between {0, 1, 2… 500} of the average load is {0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0}. The graph vector {50,100} is measured in MIPS for two types of graphs, 0.75 V when the processor is in a sleep state or hibernate, 5 V when the processor is in use when computing a task.

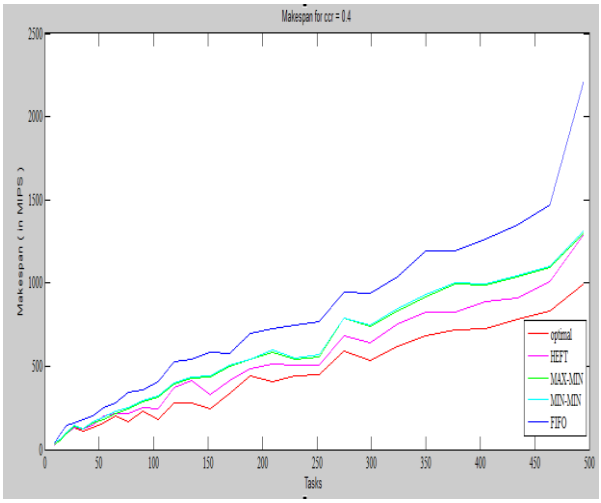
Table 2: Comparison of the results with FIFO, MINMIN, MAXMIN, HEFT and Proposed Algorithm.

Parameter	Makespan (MIPS)					
	FIFO	MINMIN	MAXMIN	HEFT	Proposed	Average Optimized Percentage
WDAG=50,ccr=0.2,Avg. Tasks=200	420.21	389.44	327.08	265.85	242.23	9.29
WDAG=50,ccr=0.4,Avg. Tasks=150	446.84	580.22	516.35	353.84	323.14	9.06
WDAG=50,ccr=0.6,Avg. Tasks=150	333.07	395.68	401.32	264.71	249.75	5.81
WDAG=50,ccr=0.8,Avg. Tasks=150	357.16	403.03	407.52	276.67	256.11	7.71
WDAG=50,ccr=1.0,Avg. Tasks=150	399.02	433.68	447.56	283.49	255.81	10.26
WDAG=50,ccr=1.5,Avg. Tasks=150	446.64	477.72	483.54	319.38	285.25	11.28
WDAG=50,ccr=2.0,Avg. Tasks=150	559.39	518.69	527.80	354.01	301.30	16.08

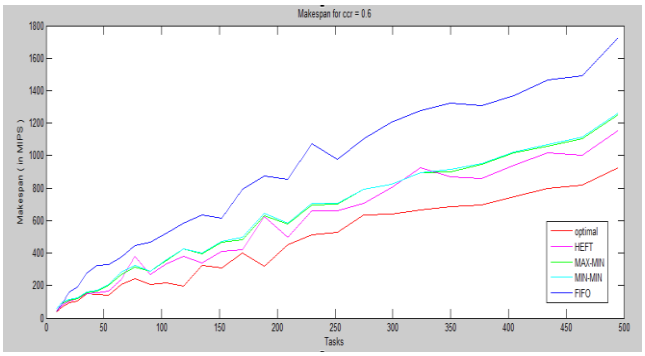
Table 2 shows the implementation results obtained for graphs for different CCR values - 0.2, 0.4, 0.6, 0.8, 1.0, 1.5 and 2.0, mean 50, HEFT, MAX MIN, MINMIN and FIFO scheduling algorithms. From Table 2, the proposed algorithm has a better percentage than the current HEFT with a minimum percentage and the weighted approach has different CCR values such as 9.29%, 9.06%, 5.81%, 7.71%, 10.26%, 11.26% and 16.08%.



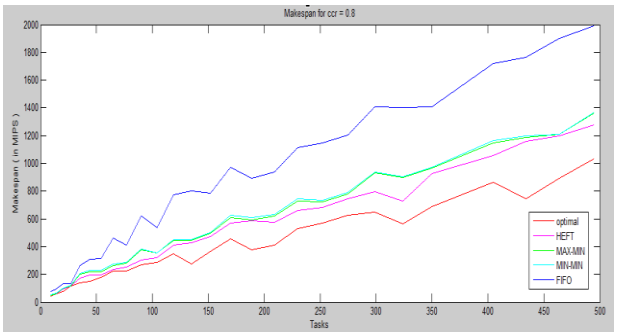
a.Makespan vs CCR=0 .2



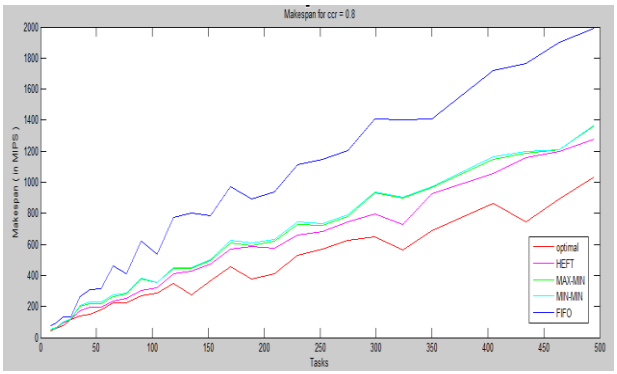
b.Makespan vs CCR=0 .4



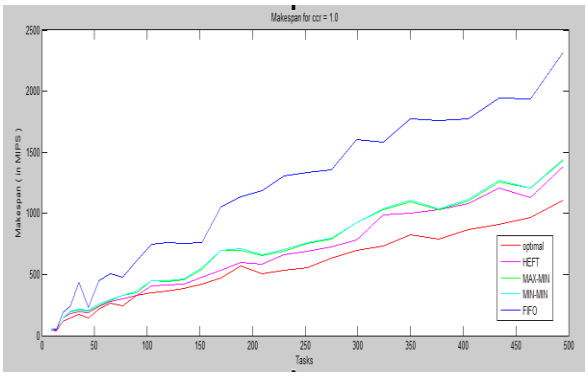
c.Makespan vs. CCR= 0.6



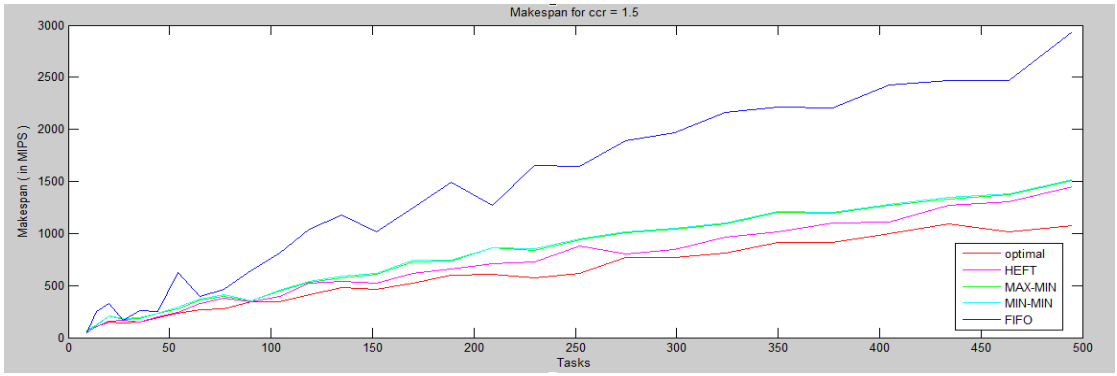
d.Makespan vs. CCR= 0.8



e.Makespan vs. CCR= 1.0



f.Makespan vs. CCR= 1.5



g.Makespan vs. CCR= 2.0

Figure 3 Graphical representation with comparison makespan various CCR values and current existing algorithms and the proposed algorithm

Figure 3 shows that the average weight of the DAG is 50, different CCR values 0.2,0,0.4,0.6,0.8,1.0 and 2.0 for an average of 150 tasks for weight graph the performance improvement of the proposed scheduling algorithm over other existing algorithms for scheduling of tasks in the cloud data center. The proposed scheduling algorithm is results percentage compare with HEFT 9.29%, 9.06%, 5.81%, 7.71%, 10.26%, 11.26% and 16.08%.

VI. CONCLUSION

This paper addresses a scheduling algorithm based on a genetic algorithm permutation-based encoding method for determining interrupted tasks before applications to reduce server slack time make-up and overall energy consumption in datacenters. The proposed algorithm improves 12% on the HEFT scheduling algorithm, 33% on the MAX-MIN scheduling algorithm, 34% on the MIN-MIN scheduling algorithm, and 42% on the FIFO scheduling algorithm. Experimental results show that our algorithm improves the HEFT algorithm, MAX-MIN scheduling algorithm, MIN-MIN scheduling algorithm and FIFO scheduling algorithm in terms of makeup. The energy consumption based on these scheduling algorithms is applied to the processor on a slack time basis and the slack time depends on the capacity of the processor, the computation cost of the tasks, and the communication cost of the tasks. Therefore, the results obtained for energy consumption depend on the type of graph, and the sluggish time on the processor is brought down compared to other algorithms. But our scheduling method gives the best results in terms of energy consumption without compromising on makespan.

VII. REFERENCES

[1] Topcuoglu, H.; Hariri, S.; Min-You Wu, "Performance-effective and low complexity task scheduling for heterogeneous computing," Parallel and Distributed Systems, IEEE Transactions on , vol.13, no.3, pp.260,274, Mar 2002.

[2] Shukla, Dhirendra Kumar, Divya Kumar, and Dharmender Singh Kushwaha. "Task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II." Materials Today: Proceedings.

[3] Mezmaz, Mohand, et al. "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems." Journal of Parallel and Distributed Computing 71.11 (2011): 1497-1508.

[4] Ying Chang-tian; Yu Jiong, "Energy-Aware Genetic Algorithms for Task Scheduling in Cloud Computing," ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh , vol., no., pp.43,48, 20-23 Sept. 2012.

[5] Tao, F., et al., CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Applied Soft Computing, 2014. 19(0): p. 264-279.

[6] Babukarthik, R.G.; Raju, R.; Dhavachelvan, P., "Energy-aware scheduling using Hybrid Algorithm for cloud computing," Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on , vol., no., pp.1,6, 26-28 July 2012.

[7] Yadav, Amrendra Singh, et al. "Increasing Efficiency of Sensor Nodes by Clustering in Section Based Hybrid Routing Protocol with Artificial Bee Colony." Procedia Computer Science 171 (2020): 887-896.

[8] Xiaomin Zhu; Huangke Chen; Yang, L.T.; Shu Yin, "Energy-Aware Rolling-Horizon Scheduling for Real-Time Tasks in Virtualized Cloud Data Centers," High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on , vol., no., pp.1119,1126, 13-15 Nov. 2013

[9] Dhirendra Kumar Shukla, Divya Kumar and Dharmender Singh Kushwaha, "An efficient tasks scheduling algorithm for batch processing heterogeneous cloud environment" in International Journal of Advanced Intelligence Paradigms. DOI: 10.1504/IJAIP.2021.10027089

[10] Dhirendra Kumar Shukla, Divya Kumar and Dharmender Singh Kushwaha, "Makespan Optimization for Dependent Task Scheduling On Multiprocessing System Using Beta ABC" Jour of Adv. Research in Dynamical & Control Systems, Vol. 11, Regular Issue 11, 2019,pp. 417-427.

[11] Zhong, X.; Cheng-Zhong Xu, "Energy-Aware Modeling and Scheduling for Dynamic Voltage Scaling with Statistical Real-Time Guarantee," Computers, IEEE Transactions on , vol.56, no.3, pp.358,372, March 2007.

[12] Alnowiser, A.; Aldhahri, E.; Alahmadi, A.; Zhu, M.M., "Enhanced Weighted Round Robin (EWRR) with DVFS Technology in Cloud Energy-Aware," Computational Science and Computational Intelligence (CSCI), 2014 International Conference on , vol.1, no., pp.320,326, 10-13 March 2014.

[13] Garg, S.K., Yeo, C.S., Anandasivam, A. and Buyya, R., 2011. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. Journal of Parallel and Distributed Computing, 71(6), pp.732-749.

[14] Yadav, Rahul Kumar Sharma1 Amrendra Singh, and Mitra Bhushan3 Mayank Deep Khare. "An Cost Effective Euclidean Steiner Tree based Mechanism for Reducing Latency in Cloud."

[15] Anton Beloglazov* and Rajkumar Buyya: Energy Efficient Resource Management in Virtualized Cloud Data Centers. 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

[16] Dhananjay Kr Sharma, Dhirendra Kr Shukla, Vijay Kr. Dwivedi, Avdhesh Kr. Gupta and Munesh C. Trivedi, "An Efficient Makespan Reducing Task Scheduling Algorithm in Cloud Computing Environment" 5 International Conference on ICT for Sustainable Development, Goa, India. (Springer) https://doi.org/10.1007/978-981-15-8354-4_31

[17] R. G. Babukarthik, R. Raju, P. Dhavachelvan: Energy-aware scheduling using Hybrid Algorithm for cloud computing. ICCCNT'12 26th_2Sdl July 2012, Coimbatore, India.

[18] Abdulaziz Alnowiser, Eman Aldhahri, and Abdulrahman Alahmadi: Enhanced Weighted Round Robin (EWRR) Scheduling with DVFS Technology in Cloud. 2014 International Conference on Computational Science and Computational Intelligence.

[19] Saurabh Kumar Garg a, ∗, Chee Shin Yeob, Arun Anandasivamc, Rajkumar Buyyaa: Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. J. Parallel Distrib. Comput. 71 (2011) 732–749.

[20] Durillo, Juan José, Vlad Nae, and Radu Prodan. "Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff."Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on. IEEE, 2013.

[21] Dhirendra Kumar Shukla, Vijay Kr. Dwivedi and Munesh C. Trivedi "Encryption Algorithm in Cloud Computing" in Materials Today Proceedings. https://doi.org/10.1016/j.matpr.2020.07.452

[22] M. Mezmaz & N. Melab & Y. Kessaci & Y.C. Lee c & E.-G. Talbi & A.Y. Zomaya & D. Tuyttens, (2011) "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems", ELSEVIER, J. Parallel and Distributed Computing.

[23] Ying Chang-tian; Yu Jiong, "Energy-Aware Genetic Algorithms for Task Scheduling in Cloud Computing," ChinaGrid Annual Conference (ChinaGrid), 2012 Seventh , vol., no., pp.43,48, 20-23 Sept. 2012

[24] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multi objective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[25] T.L. Adam, K.M. Chandy, and J. Dickson, "A Comparison of List Scheduling for Parallel Processing Systems," Communications of the ACM, vol. 17, Dec. 1974, pp. 685-690.

[26] Juarez, F., Ejarque, J. and Badia, R.M., 2018. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Future Generation Computer Systems, 78, pp.257-271.

[27] S.Sindhu, Dr.Saswati Mukherjee: A Genetic Algorithm based Scheduler for Cloud Environment. 2013 International Conference on Computer and Communication Technology (ICCCT).

[28] Chun-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang, and Chu-Sing Yang: A Hyper-Heuristic Scheduling Algorithm for Cloud. IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 2, APRIL-JUNE 2014