Adv Syst Sci Appl 2024; **02**:192–202
Published online at https://ijassa.ipu.ru.

A Generalized Bi-Objective Scheduling Algorithm for Batch-of-Tasks on Heterogeneous Computing Systems

Mohammad Qasim[1], Mohammad Sajid[1], Maria Lapina[2], Mikhail Babenko[2*]

[1]*Aligarh Muslim University, Aligarh, India*
[2]*North-Caucasus Federal University, Stavropol, Russia*

Abstract: The high energy consumption of data centers and its contribution towards greenhouse gases demand energy-efficient management of resources. Energy consumption of computing resources encourages the development of bi-objective scheduling algorithms optimizing the makespan of jobs and energy consumption of computing resources. In general, the problem of job scheduling and bi-objective optimization falls in the NP-complete combinatorial optimization problem category. To address the bi-objective scheduling problem, a generalized bi-objective scheduling algorithm (Z*) for Batch-of-Tasks (BoT) applications on the Heterogeneous Computing System (HCS) has been proposed. The BoT represents the set of independent tasks from multiple applications, and the HCS represents the computational environment consisting of processors with different frequencies. To schedule tasks, the Z* algorithm takes decisions using the optimization function of energy consumption and completion time of tasks based on the given weights. The weight could be fractional or integer, so the Z* algorithm represents a set of different algorithms. The proposed algorithm is beneficial for cloud data centers/service-oriented computing to execute customer jobs based on the demand, whether the customer needs high throughput or low cost of execution.

Keywords: Batch-of-Tasks, Heterogeneous Computing System, Scheduling, Makespan, Energy

1. INTRODUCTION

On one side, the performance of data centers has been improved by increasing the parallel servers, but on the other side, the overall energy consumption of data centers housing thousands of servers needs to be minimized due to negative environmental effects [1, 2]. For example, a data center houses the ten fastest systems (Tianhe-2) to provide services to users. Since a Tianhe-2 server requires 17.808 MW of power, making a single hour's execution on Tianhe-2 cost roughly $1780.80 on the assumption that the average price of electricity is $100/MW-Hour. If the cloud data service is available for 8 hours per day, 20 days per month, the cost of operating a cloud data center apart from the cooling cost will be $34 million per year. To provide power to the data center, the powerhouse needs to consume 71230 kg coal and 712320L water while producing 177545.8 kg CO2 and 5342.4 kg SO2 [1–5]. Furthermore, high energy use raises temperature since each electronic component's temperature has a direct inverse relationship with its energy use, which lowers system reliability. According to NRDC [2], In 2013, it was predicted that 12 million computing servers across 3 million data centers in the US used 91 billion kilowatt-hours of electricity. By 2020, this energy usage is projected to increase by a direct 53%, reaching 140 billion

[*]Corresponding author: mgbabenko@ncfu.ru

kilowatt-hours. Due to the high financial costs, decreased reliability of computer systems, and adverse impacts on the environment, the excessive energy usage of data centers has become a concern, which encourages the development of energy-efficient hardware and software management [6].

Heterogeneous Computing Systems (HCSs) are the base of data centers due to the replacement and up-gradation of computing nodes. Compared to homogeneous computing, HCS can be thought of as the coordinated employment of a variety of processing units with varying capabilities to give higher performance, better energy efficiency, and more cost-effectiveness [7]. To deploy energy efficiency, energy-aware scheduling models are required to develop that execute jobs with minimum possible energy consumption as well as makespan. Scheduling generally involves distributing tasks among processors with the purpose of optimizing one or more quality of service criteria. To find the task-to-processor mapping that minimizes both makepan and energy usage, scheduling methods are necessary. Since data centers execute jobs in three categories, i.e., lower cost of execution with no constraints on time, lower execution time with no constraint on cost and the combination of both. The cost of execution is measured in terms of direct energy consumption and the time span holding the resources of data centers. Therefore, the data centers require a bi-objective scheduling algorithm that has the capability to offer different weights to both execution time and energy consumption/cost. In this work, a generalized bi-objective scheduling algorithm (Z*) for Batch-of-Tasks (BoT) on a HCS has been proposed. The BoT is a very common job representing the set of independent tasks from multiple applications, and HCS represents the computational environment consisting of processors with different fixed frequencies. The proposed Z* algorithm schedules tasks based on a weighted aggregation of energy consumption and completion time of tasks.

The remainder of the paper is organized as follows. The next section explains the related scheduling algorithm that schedules BoT on HCS. The design principles, which include the BoT model, the HCS model, and the energy consumption model, are presented in Section 3. The proposed generalized Z* algorithm and its optimization function are described in Section 4. Section 5 shows that Z* is the same as some of the existing algorithms, and the simulation results present the behavior for different cases. The paper ends in Section 6 with concluding remarks and possible future directions.

2. RELATED WORK

The standard methods for reducing the makespan of BoT are MaxMin [10, 11], MinMin [10, 11], MaxMin+ [12], MinMin+ [12], duplication algorithms [14], analytical models [15] and dependent task algorithms [15]. The Expected Time to Compute (ETC) simulation model provides a standard by which to compare the performance of these algorithms on a distributed computing platform. Several of the algorithms are described in the following list.

The MaxMin algorithm operates in two stages as well. In the first stage, it determines the shortest time to complete each unscheduled task, considering all processors. In the second stage, the task that will take the longest to finish overall is selected and assigned to the appropriate processor. This process is repeated until all tasks have been scheduled, at which point the newly scheduled item is removed from the list of unscheduled tasks [10, 11].

The MinMin algorithm comprises two stages. It determines the minimal completion time for every unscheduled task in the first Stage while considering all processors. The job that takes the shortest time to complete overall is selected and allocated to the suitable processor in the second stage. Once a task has been scheduled, it is deleted from the list of unscheduled tasks, and the process is then repeated until every item has been scheduled [10, 11].

Another form of a MinMin algorithm is known as MinMin+ with less time complexity. Instead of being task-oriented like in the MinMin, and MinMin+ algorithm operates from a processor-oriented perspective. After initialization, the MinMin+ algorithm determines the completion times of each task with regard to each processor. After that, algorithm runs in two phases. The algorithm initially identifies the task that will take each processor the fewest amount of time to complete. The task that will take the least amount of time to complete will be scheduled and taken out of the set. Since the current task's scheduling decision only influences the single processor's readiness time, this approach only calculates the time requirements of all unscheduled tasks with regard to the same single processor. This avoids calculating the completion times of tasks with respect to all other processors. The two-stage process continues until all tasks have been scheduled [12].

There have been proposed a plethora of bi-objective algorithms with the aim of optimizing energy consumption as well as makespan. Some of the algorithms that are energy-aware are Global Scheduling with Shared Slack Reclamation (GSSR) [17], Fixed-order list scheduling with Shared Slack Reclamation (FLSSR) [17], Shortest Task First for Computer with Minimum Energy (STF-CME) algorithm [18], Longest Task First for Computer with Minimum Energy (LTF-CME) algorithm [18], MaxMax min [19], MinMean min [19], MINMin [20], MINSuff [20], MinMIN-MinMin [20], kPB-Sufferage [20], G2 [21], ESTS algorithm [22], Min-Energy algorithm and ESHEFT algorithm [24].

To minimize energy consumption while meeting deadlines for all the tasks, Zhang et al. presented several energy-efficient algorithms, such as the Shortest Task First for Computer with Minimum Energy (STF-CME) algorithm and the Longest Task First for Computer with Minimum Energy (LTF-CME) algorithm. The initial stage of each algorithm schedules tasks to the machine, and the final stage establishes the ideal or nearly ideal speeds for each task. According to simulation research, hybrid algorithms outperform and provide the optimum job schedules for both discrete and continuous-speed computers [18].

For independent tasks, Diaz et al. suggested three low-cost energy-aware algorithms: MaxMax min, MinMax min and MinMean min. The two phases of the low-cost heuristic's operation. The method gives each task a priority in the first phase, sorts the tasks into decreasing priority order, and then uses the ordered task's list to choose the best processor. In the next phase, calculate the Scoring Function (SF) for the chosen task relative to each processor and scheduling the work for the processor with the lowest SF value. Up till all the tasks are scheduled, the process is repeated [19].

Twenty bi-objective energy-conscious scheduling methods were proposed and assessed by Nesmachnow et al. [20] utilizing the algorithms MaxMin, MinMin, Suffrage, and k-Percent Best (kPB). Based on a heterogeneous multi-core environment, the proposed algorithms are assessed—MINMin functions in two stages. In the first stage, it chooses task and processor combinations that, when all processors are considered, have the shortest completion times for each unscheduled task. The work is allocated to the relevant processor in the second stage using pair from the stage 1 pairs that minimize energy consumption. The most recent task scheduled is then removed from the list of unscheduled tasks; this process is repeated for each task that is still on the list until all tasks have been scheduled [20].

Sajid et al. propose two bi-objective energy-aware scheduling algorithms using the weighted aggregation method to schedule BoT applications on heterogeneous processors with the objective of optimizing makespan and energy consumption. The proposed algorithms employ processor-oriented to schedule the tasks and hence, offer lower time complexity in comparison to MinMin, MaxMin, MINSuff and MINMin algorithms [10, 11, 20]. The effectiveness of the suggested algorithms has been compared with MaxMin, MinMin, MINMin and MINSuff using the ETC simulation model. The simulation study reveals that the proposed algorithm (G2) performs better than peer algorithms in terms of flowtime. energy consumption and makespan [21].

In this work, we propose a generalized algorithm Z* that offers the selection of weights is flexible to energy consumption and/or completion times. It is a useful algorithm to compare the different versions of the algorithms.

3. THE PROBLEM FORMULATION

In this section, the problem is formulated, including BoT model, a HCS, design principles and energy consumption [21].

3.1. *Batch-of-Tasks and Heterogeneous Computing System*

The BoT B consists of N tasks and is represented by $B = t_1,\ t_2 \ldots\ldots t_i \ldots\ t_N$. Every task $t_i \in B$ is independent and atomic in nature, i.e., the tasks have no precedence constraints and cannot be partitioned/ distributed/pre-empted during execution. The expected number of cycles $(CYCLE_i)$ required to execute a task $t_i \in B$ is also given. The BoT applications have been used by many streams, such as high energy physics, geophysics, molecular quantum mechanics, weather forecasting, climate, cryptographic analysis, and tomographic reconstruction. The HCS is explained in the next paragraph and Figure 3.1. shows the mechanism of HCS in cloud environments.

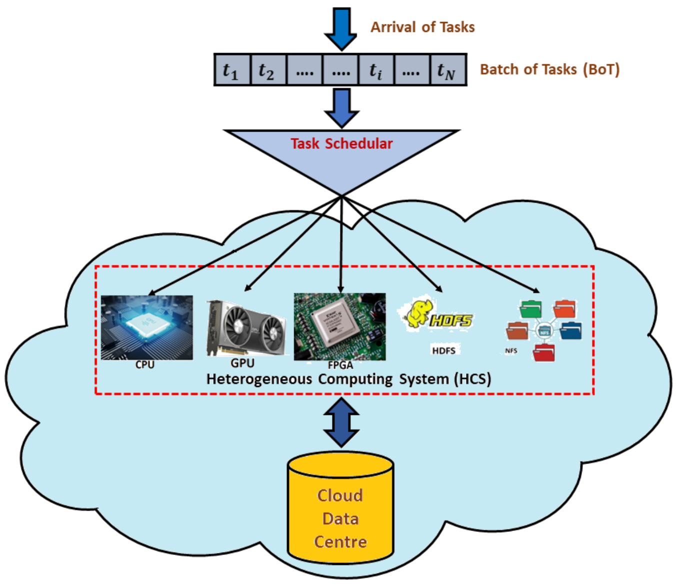

Fig. 3.1. Heterogeneous Computing in Cloud Environments.

The HCS H represents the execution environment consisting of M heterogeneous processors and is characterized by $H = p_1, p_2 \ldots\ldots p_j \ldots p_M$. All processors are heterogeneous, i.e., the frequency f_j and voltage v_j of the processor $p_j \in H$ is different from the processor $p_k \neq p_j$. A high-speed interconnection network connects all processors, e.g., Myrinet, to execute real-time tasks. The processors are not dynamic voltage-frequency scaling-enabled, and each processor $p_j \in H$ runs on fixed voltage (v_j) with corresponding fixed frequency (f_j).

3.2. *Design Principles*

The scheduler maintains information about a scheduler node, which assigns the BoT B of N independent tasks to the HCS H of M heterogeneous processors and manages the batch of independent jobs on a periodic basis. In the sequence of their execution, all tasks assigned to such a processor p_j is en-queued in a local queue Q_j. We employ the following notations and design ideas in this work.

ETC [N][M]: The proposed algorithm has been assessed using the simulation model known as Expected-Time-to-Compute (ETC). The estimated execution time for every task on each processor [10,11,20] is stored in a matrix called ETC[N][M] that is ordered N-by-M.

It is possible to calculate the task t_i expected runtime on the processor p_j using frequency fj as follows:

$$ETC\,(i,j) = \frac{CYCLE_i}{f_j}, \tag{3.1}$$

ALLOC [N][M]: The distribution of N tasks across M processors for execution is represented by an allocation matrix (ALLOC) of order N by M. Whether or not the task t_i has been assigned to the processor p_j determines whether the value of ALLOC (i,j) is zero or one. The allocation matrix should satisfy the following restrictions after scheduling N tasks.

$$\sum_{J=1}^{M} ALLOC(i,j) = 1 \quad \forall 1 \le i \le N \tag{3.2}$$

$$\sum_{i=1}^{N} ALLOC(i,j) = K, \quad \forall 0 \le K < N, \quad \forall 1 \le j \le M \tag{3.3}$$

PRT_j: It indicates the amount of time that the processor p_j has been ready. The PRT_j for the processor p_j is the earliest time processor p_j has the ability to finish all of the tasks that were previously scheduled for completion. If there is no task allocated to the processor p_j, the $PRT_j = 0$.

RT_{ij}: It indicates the task t_i ready time on the processor p_j. RT_{ij} is the processor-ready time for the processor p_j on which task t_i is scheduled i.e.

$$RT_{ij} = PRT_j \tag{3.4}$$

CT_{ij}: It shows when a task t_i on the processor p_j was completed. It may be expressed mathematically as the summation ready time of the processor p_j and the expected time it will take to complete the task t_i.

$$CT_{ij} = PRT_j + \ ETC\,(i,\ j) \tag{3.5}$$

After scheduling the task t_i on processor p_j, the processor ready time is also updated as

$$PRT_j = CT_{ij} \tag{3.6}$$

Makespan of BoT B: It is referred to as the maximum amount of time that elapsed between the first scheduled task's start time and the last scheduled task's completion time of the batch of N tasks, and it may be calculated as

$$MK_{B,H} = max\,\{CT_{ij} | ALLOC(i,j) = 1, 1 \le i \le N, \} \tag{3.7}$$

3.3. Heterogeneous Computing Systems Energy Model

To execute the BoT B on HCS H, the processors consume energy deepening on the schedule and order of tasks. The CMOS-fabricated processor-based system's energy comprises of Static Energy (E_{static}) and Dynamic Energy ($E_{dynamic}$) with the significant energy consumption factor being $E_{dynamic}$. For a specific frequency f_j and voltage v_j, the dynamic power ($P_{dynamic}$) used by a processor p_j is given by [24].

$$P_{Dynamic}\,(p_j) = \quad a * C_j * {v_j}^2 * f_j \tag{3.8}$$

where the activity factor is represented by the constant α, C_j represented physical capacitance, processor's p_j clock frequency (speed) denoted by f_j and V_j denotes the voltage applied to the processor corresponds to the frequency f_j. The physical capacitance (C_j) and activity factor (α) is determined during the manufacturing and design phases. The supply voltage V_j and the clock frequency f_j are configurable either statically (when compiling) or dynamically (when running).

Due to the heterogeneity of the processors utilized, the energy consumption of a task t_i is dependent on the frequency of operation, voltage, and the amount of time necessary to complete the task. Consequently, a task's t_i energy consumption on a processor p_j may be expressed as

$$EC_{ij} = a * C_j * {V_j}^2 * f_j * ETC\,(i, j) \tag{3.9}$$

The energy consumed by BoT B on HCS H is given by

$$E_{B,\,H} = \sum_{i=1}^{N}\sum_{j=1}^{M} ALLOC\,(i, j) * EC_{ij} \tag{3.10}$$

4. THE GENERALIZED BI-OBJECTIVE ALGORITHM (Z*)

In this section, the proposed generalized bi-objective algorithm (Z*) is explained. The Z* algorithm is a static scheduling algorithm to schedule a BoT of N independent tasks on HCS H of M heterogeneous processors. The objective of the algorithm is to optimize either makespan or energy or both depending on the weights assigned to objectives. The Z* algorithm represents the set of 25 algorithms for integer weights and a set of infinite algorithms based on fraction weights. The algorithms A1 to A25 are given in Table 4.1. The explanation of the Z* algorithm is given as follows.

4.1. Optimization Function Specification

The BoT B of N independent tasks results in negligible idle slots while scheduling on an HCS H of M processors [21]. The Optimization Function (OF) employed by the Z* algorithm does not consider the idle; however, it considers both the energy consumption and completion time of tasks as follows.

Completion time of task – As defined in equation (3.5).

Energy Consumption of task – As per equation (3.9).

The weighted sum approach, which is commonly used in the literature, has been applied for the optimization function (OF), combining both objectives into one objective by multiplication of weights for each objective [8]. The drawback of this optimization technique is also widely known: only solutions found in the convex region of the Pareto front (a group of solutions to the problem that are not dominated by any other solutions), and only one solution from the Pareto front is discovered in each run. However, the solution obtained using the weighted sum method is always Pareto-optimal if the weights are positive for all objectives.

The definition of the OF based on the normalized values of the goals measured in the same unit (constant). Then, given task t_i on the processor p_j, the Normalized Energy $(NE(t_i, p_j))$ and the Normalized Completion Time $(NC(t_i, p_j))$ can be calculated as

$$NE\,(t_i, p_j) = \frac{EC\,(i,\ j)}{max(EC\,(i,\ k))} \quad \forall p_k \in P \tag{4.11}$$

$$NC\,(t_i, p_j) = \frac{CT_{ij}}{max(CT_{ik})} \quad \forall p_k \in P \tag{4.12}$$

It is obvious that $0 < NE(t_i, p_j) \leq 1$, $0 < NC(t_i, p_j) \leq 1$. The Optimization Function $(OF(t_i,\ p_j))$ can then be written using (11) and (12) as

$$OF\left(t_i, p_j\right) = \beta_1 * NC\left(t_i, p_j\right) + \ \beta_2 * NE\left(t_i, p_j\right) \tag{4.13}$$

The optimization function is computed for the task t_i considering all processors followed by the scheduling decisions. The performance and priority of the Z* algorithm depend on the constants (β_1, β_2) of an optimization function that integrates both objectives and based on the value, attempts to optimize both. Therefore, the values of constants (β_1, β_2) are critical and must be chosen based carefully depending on the requirements. The constants $beta_1$ and $beta_2$ must be between –1 to +1, i.e., $\beta_1 \in [-1, 1]$, $\beta_2 \in [-1, 1]$ and must fulfil the following two constraints

$$|\beta_1| + \ |\beta_2| = 1 \tag{4.14}$$

$$|\beta_1| + \ |\beta_2| \neq 0 \tag{4.15}$$

Here, constraint (4.14) states that the weights of the objectives cannot be zero, and the sum of weights assigned to objectives must be one. For positive values of β_1 and β_2, the $OF(t_i, p_j)$ minimizes the objectives, whereas it maximizes the objectives for negative values of β_1 and β_2.

4.2. The Procedure of the Z Algorithm*

Algorithm 1 provides the pseudo code for the Z* algorithm.

Algorithm 1: Pseudo Code of Z*

```
Input: HCS H of M processors, BoT B of N tasks
Output: Schedule $E_{B,H}, MK_{B,H}, S$
1  Generate ETC using equation (3.1)
2  Assign values to $\beta_1$ and $\beta_2$ for optimization function
3  while $B \neq \phi$ do
4      $ST = \phi$ // The pair of task and processors that have the lowest cost functions
5      for $\forall t_i \in H$ do
6          for $\forall p_i \in H$ do
7              Calculate $NE(t_i, p_j)$ by using eq (4.11)
8              Calculate $NC(t_i, p_j)$ by using eq (4.12)
9              Calculate $OF(t_i, p_j)$ by using eq (4.13) for specidied values of $\beta_1$ and $\beta_2$
10         end
11         Find pair $(t_i, p_j)$ with min value $OF(t_i, p_j)$
12         Add pair $(t_i, p_j)$ to $ST$ i.e. $ST = ST \cup (t_i, p_j)$
13     end
14     Find pair $(t', p_k)$ from ST with min. value of $OF(t', p_k)$
15     Set value in ALLOC matrix corresponding to $(t', p_k)$
16     Allocate task $t'$ on processor $p_k$
17     Eliminate task $t'$ from B i.e. $B = B - t'$
18 end
19 Find schedule S using ALLOC matrix
20 Calculate $MK_{B,H}$ using (3.7)
21 Calculate $E_{B,H}$ using (3.10)
22 Return Schedule $E_{B,H}, MK_{B,H}, S$
```

First of all, the Z* algorithm assigns values to their objectives, i.e., β_1 and β_2 and then proceed to schedule the tasks using a while loop from steps 3 to 18. The Z* algorithm computes the optimization function $(OF(t_i, p_j))$ for each unscheduled task against all processors using equation (4.13) and determines the pair (t_i, p_j) of task t_i and processor pj where processor pj offers a minimum value of optimization function. Next, the algorithm finds the pair (t', p_k) that provides the lowest value for the optimization function $(OF(t', p_k))$. The task t' has now been assigned to p_k and it has been withdrawn from BoT B. This process is then repeated for the remaining tasks until they have all been assigned to a processor.

4.3. Z for Integer Weights of β_1 and β_2*

For different values of β_1 or β_2 with the constraints in equations (4.14) and (4.15), Z* represents a set of 25 algorithms A1 to A25 listed in Table 4.1; some of them are already well-known and established algorithms, viz. MinMin [10, 11], MaxMin [10, 11], MINMin [20], MinMIN [20], MINMIN [20] and MaxMIN [20]. Table 4.1 depicts 25 algorithms from A1 to A25. The A1, A2, A3, A11, A12 and A13 behave similarly to MinMin, MaxMin, MINMin, MinMIN, MaxMIN and MINMIN, respectively.

Table 4.1. Different forms of Z* algorithm

Sr. No.	Phase I		Phase II		
	β_1	β_2	β_1	β_2	Algorithm
1	1	0	1	0	A1 or MinMin [10,11]
2	1	0	-1	0	A2 or MaxMin [10,11]
3	1	0	0	1	A3 or MINMin [20]
4	1	0	0	-1	A4
5	1	0	(-1, 1)	(-1, 1)	A5
6	-1	0	1	0	A6
7	-1	0	-1	0	A7
8	-1	0	0	1	A8
9	-1	0	0	-1	A9
10	-1	0	(-1, 1)	(-1, 1)	A10
11	0	1	1	0	A11 or MinMIN [20]
12	0	1	-1	0	A12 or MaxMIN [20]
13	0	1	0	1	A13 or MINMIN [20]
14	0	1	0	-1	A14
15	0	1	(-1, 1	) (-1, 1)	A15
16	0	-1	1	0	A16
17	0	-1	-1	0	A17
18	0	-1	0	1	A18
19	0	-1	0	-1	A19
20	0	-1	(-1, 1)	(-1, 1)	A20
21	(-1, 1)	(-1, 1)	1	0	A21
22	(-1, 1)	(-1, 1)	-1	0	A22
23	(-1, 1)	(-1, 1)	0	1	A23
24	(-1, 1)	(-1, 1)	0	-1	A24
25	(-1, 1)	(-1, 1)	(-1, 1)	(-1, 1)	A25

4.4. Z for Fractional Weights of β_1 and β_2*

It is very obvious that the weights of β_1 and β_2 between –1 to +1, there exist infinite weights for β_1 and β_2. Therefore, the Z* algorithm represents a set of infinite algorithms for different fractional weights.

4.5. Z as MinMin*

The proof that Z* behaves like MinMin is given. The proof for MaxMin, MINMin, MinMIN, MaxMIN and MINMIN are similar. The outcome of the Z* depends on the values of β_1 and β_2. For instance, it behaves similarly to the MinMin algorithm (A1) if the values assigned to β_1 and β_2 are 1 and 0, respectively, for both phases. Since the Normalized Completion Time metric $(NC(t_i, p_j))$ for task t_i on processor p_j is the ratio of the completion time of a task t_i on processors p_j $(\mathrm{CT}_{ij}))$ and a maximum of the completion times of the task t_i against each processor p_k $(1 \leq k \leq M \, or \, p_k \in P)$. If $\beta_1 = 1$ and $\beta_2 = 0$, then the Optimization function

$(OF(t_i, p_j))$ in phase I minimizes the makespan of the tasks and generates pair of tasks and processors (t_i, p_j) such that p_j offers minimum completion time to task t_i. It is similar to phase I of the MinMin algorithm. Again, if $\beta_1 = 1$ and $\beta_2 = 0$, then the Optimization function $(OF(t_i, p_j))$ phase II minimizes the makespan of the tasks. Phase II of the Z* algorithm considers the pairs generated in phase I. It determines the pair of tasks and processors (t_k, p_l) such that p_l offers minimum completion time to task t_k among all the pairs. It is similar to phase II of the MinMin algorithm.

4.6. Use of Z in Data Centers*

There are many ways to employ the Z* algorithm based on the requirements. The Z* algorithm can be used to:

1. Minimize only completion time for higher throughput of data centers to generate more monetary benefits.
2. Minimize only the energy consumption for lower negative environmental effects and cooling costs.
3. Minimize only completion time for foreground jobs and minimize energy consumption for background jobs.
4. As a simulation model to compare the performance for different weights.

5. CONCLUSION

The energy consumption of data centers is a considerable issue from an environmental point of view due to GHG gas emissions and from the service provider's view due to the extra burden of cooling costs. Therefore, energy-efficient of resources at data centers is demanding. In this work, we proposed a generalized energy and makespan algorithm for Batch-of-Task (BoT) on a Heterogeneous Computing System (HCS) which has the capability to minimize either the makespan, energy or both. The proposed Z* algorithm schedules tasks based on a newly defined optimization function, and it represents the set of different algorithms based on the weights of objective functions. The Z* algorithm can be employed by data centers in different ways, either for fast execution or low energy consumption, depending on the job types.

This work can be further extended to a heterogeneous system of DVFS-enabled processors having different speed and voltage levels. This work can also be realized for a single dependent job, a batch of dependent jobs, open shop scheduling problems, as well as job shop scheduling problems. Further, the proposed algorithm Z* can be further extended for jobs having stochastic execution times.

ACKNOWLEDGEMENTS

This work was supported by the Russian Science Foundation 19-71-10033, https://rscf.ru/project/19-71-10033/

REFERENCES

1. Meuer, H., Strohmaier, E., Dongarra, J., Simon, H. & Meuer, M. (2023). *Top 500 Supercomputers 2014*, [Online]. Available: http://www.top500.org/lists/2014/11/.
2. Nartural Resources Defense Council (2023). *Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barrier, Natural Resources Defense Council 2014*, [Online]. Available https://infrastructureusa.org/scaling-up-energy-efficiency-across-the-data-center-industry/.

3. Bilal, K., Malik, S. U. R., Khan, S. U., & Zomaya, A. Y. (2014). Trends and challenges in cloud datacenters. *IEEE cloud computing*, **1**(1), 10–20.
4. Sajid, M., & Raza, Z. (2013, December). Cloud computing: Issues & challenges. *In International conference on cloud, big data and trust*, **20**(13), 13–15
5. Zhua, X., He, C., Li, K., & Qin, X. (2012). Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters. *Journal of parallel and distributed computing*, **72**(6), 751–763.
6. Venkatachalam, V., & Franz, M. (2005). Power reduction techniques for microprocessor systems. *ACM Computing Surveys (CSUR)*, **37**(3), 195–237.
7. Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., et. el. (2006). The landscape of parallel computing research: A view from berkeley. University of California, Berkeley, 1–54.
8. Coello, C. A. C. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer. com.
9. Leung, J. Y. (Ed.). (2004). *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press.
10. Freund, R. F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., et. el. (1998, March). Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. *In Proceedings Seventh Heterogeneous Computing Workshop*, 184–199.
11. Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., et. el. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed computing*, **61**(6), 810–837.
12. Tabak, E. K., Cambazoglu, B. B., & Aykanat, C. (2013). Improving the performance of independenttask assignment heuristics minmin, maxmin and sufferage. *IEEE Transactions on Parallel and Distributed Systems*, **25**(5), 1244–1256.
13. Sajid, M., & Raza, Z. (2016). Turnaround time minimization-based static scheduling model using task duplication for fine-grained parallel applications onto hybrid cloud environment. *IETE Journal of Research*, **62**(3), 402–414.
14. Sajid, M., & Raza, Z. (2012, December). Level based task duplication strategy to minimize the job turnaround time. *In 2012 2nd IEEE International Conference on Parallel*, Distributed and Grid Computing, 164–169.
15. Sajid, M., & Raza, Z. (2015). An analytical model for resource Characterization and parameter Estimation for DAG-based jobs for homogeneous systems. *International Journal of Distributed Systems and Technologies (IJDST)*, **6**(1), 34–52.
16. Shahid, M., Raza, Z., & Sajid, M. (2015). Level based batch scheduling strategy with idle slot reduction under DAG constraints for computational grid. *Journal of Systems and Software*, 108, 110–133.
17. Zhu, D., Melhem, R., & Childers, B. R. (2003). Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *IEEE transactions on parallel and distributed systems*, **14**(7), 686–700.
18. Zhang, L. M., Li, K., Lo, D. C. T., & Zhang, Y. (2013). Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds. *Sustainable Computing: Informatics and Systems*, **3**(2), 109–118.
19. Diaz, C. O., Guzek, M., Pecero, J. E., Danoy, G., Bouvry, P., et. el. (2011, July). Energy-aware fast scheduling heuristics in heterogeneous computing systems. *In 2011 International Conference on High Performance Computing & Simulation*, 478–484.
20. Nesmachnow, S., Dorronsoro, B., Pecero, J. E., & Bouvry, P. (2013). Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of grid computing*, 11, 653–680.
21. Sajid, M., Raza, Z., & Shahid, M. (2016). Energy-efficient scheduling algorithms for batch-of-tasks (BoT) applications on heterogeneous computing systems. *Concurrency and Computation: Practice and Experience*, **28**(9), 2644–2669.

22. Li, K., Tang, X., & Li, K. (2013). Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, **25**(11), 2867–2876.
23. Oxley, M. A., Pasricha, S., Maciejewski, A. A., Siegel, H. J., Apodaca, J., et. el. (2014). Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system. *IEEE Transactions on Parallel and Distributed Systems*, **26**(10), 2791–2805.
24. Sajid, M., & Raza, Z. (2016). Energy-aware stochastic scheduling model with precedence constraints on DVFS-enabled processors. *Turkish Journal of Electrical Engineering and Computer Sciences*, **24**(5), 4117–4128.