electronics

MDPI

Article

A Cost-Optimized Data Parallel Task Scheduling with Deadline Constraints in Cloud

N. R. Rajalakshmi [1], Ankur Dumka [2,3], Manoj Kumar [4], Rajesh Singh [5], Anita Gehlot [5], Shaik Vaseem Akram [5,6], Divya Anand [7,8,*], Dalia H. Elkamchouchi [9] and Irene Delgado Noya [8,10]

1 Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai 600062, India; rajirajasekaran@gmail.com
2 Department of Computer Science and Engineering, Women Institute of Technology, Dehradun 248007, India; ankurdumka2@gmail.com
3 Department of Computer Science and Engineering, Graphic Era Deemed to Be University, Dehradun 248007, India
4 School of Computer Science & Engineering, Shri Mata Vaishno Devi University, Katra (J&K) 182320, India; vermamk@gmail.com
5 Division of Research & Innovation, Uttaranchal Institute of Technology, Uttaranchal University, Dehardun 248007, India; drrajeshsingh004@gmail.com (R.S.); dranitagehlot@gmail.com (A.G.); vaseemakram5491@gmail.com (S.V.A.)
6 Law College of Dehradun, Uttaranchal University, Dehradun 248007, India
7 School of Computer Science & Engineering, Lovely Professional University, Phagwara 144411, India
8 Higher Polytechnic School, Universidad Europea del Atlantico, C/Isabel Torres 21, 39011 Santander, Spain; irene.delgado@uneatlantico.es
9 Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia; dhelkamchouchi@pnu.edu.sa
10 Department of Project Management, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
* Correspondence: divyaanand.y@gmail.com

Abstract: Large-scale distributed systems have the advantages of high processing speeds and large communication bandwidths over the network. The processing of huge real-world data within a time constraint becomes tricky, due to the complexity of data parallel task scheduling in a time constrained environment. This paper proposes data parallel task scheduling in cloud to address the minimization of cost and time constraints. By running concurrent executions of tasks on multi-core cloud resources, the number of parallel executions could be increased correspondingly, thereby, finishing the task within the deadline is possible. A mathematical model is developed here to minimize the operational cost of data parallel tasks by feasibly assigning a load to each virtual machine in the cloud data center. This work experiments with a machine learning model that is replicated on the multi-core cloud heterogeneous resources to execute different input data concurrently to accomplish distributive learning. The outcome of concurrent execution of data-intensive tasks on different parts of the input dataset gives better solutions in terms of processing the task by the deadline at optimized cost.

Keywords: data parallel task; virtual machine; cloud data center; cost optimization model; concurrent computation

Citation: Rajalakshmi, N.R.; Dumka, A.; Kumar, M.; Singh, R.; Gehlot, A.; Akram, S.V.; Anand, D.; Elkamchouchi, D.H.; Noya, I.D. A Cost-Optimized Data Parallel Task Scheduling with Deadline Constraints in Cloud. *Electronics* **2022**, *11*, 2022. https://doi.org/10.3390/electronics11132022

Academic Editor: Shinichi Yamagiwa

Received: 25 May 2022
Accepted: 21 June 2022
Published: 28 June 2022

1. Introduction

The development of software is an enormous sector of the global economy, with its own phase of evolution and a significant influence on the digital economy as a whole. The IT sector is a real engine of growth in the world economy, which means that its success is important. The escalation of computing services has been enhanced recently. Its attractiveness mainly stems from the release of IT resources, such as the transformation of capital IT expenditure into economic resources. It has the potential to minimize costs by economies of scale. Cloud computing, big data, and the Internet of Things (IoT) have

become inseparable components of digital networking and information systems nowadays. Along with numerous other sources of information and communication, they cover different facets of society. Total automation and a focus on observable processes provide a continuous flow of digital data that can be used at various levels of management, business growth, manufacturing processes of an organization, and even the perception of the software product by the customer. As an unprecedented volume of data is being dealt with by IT industries every day, these vast amounts of data create new challenges and opportunities.

It has been affirmed that the execution of gathered massive, non-uniform real-world data through grid computing and cloud computing is becoming more obscure [1]. The challenges are not bound to the size of data only but also to the time and cost constraints of execution [2]. Task scheduling approaches have been used for homogeneous multiprocessor systems to solve computationally intensive image processing and computer vision applications, including those for reconfigurable network topologies. However, in contrast with homogeneous multiprocessor systems, heterogeneity in computer systems adds an additional degree of complexity to the scheduling problem. This could be overcome by efficient scheduling of tasks because the optimal scheduling uses the resources efficiently to achieve quick response times for real-time applications [3]. The scheduling of heterogeneous computing resources depends on the following parameters of resource availability, workload size, resource utilization, cost, and resource capacity, such as different speeds of processors and communication between the processors. Service level agreement (SLA) negotiation also moderates the scheduling and utilization of resources [4]. Therefore, the above parameters have to be considered while scheduling the tasks on resources to meet user expectations (Figure 1).

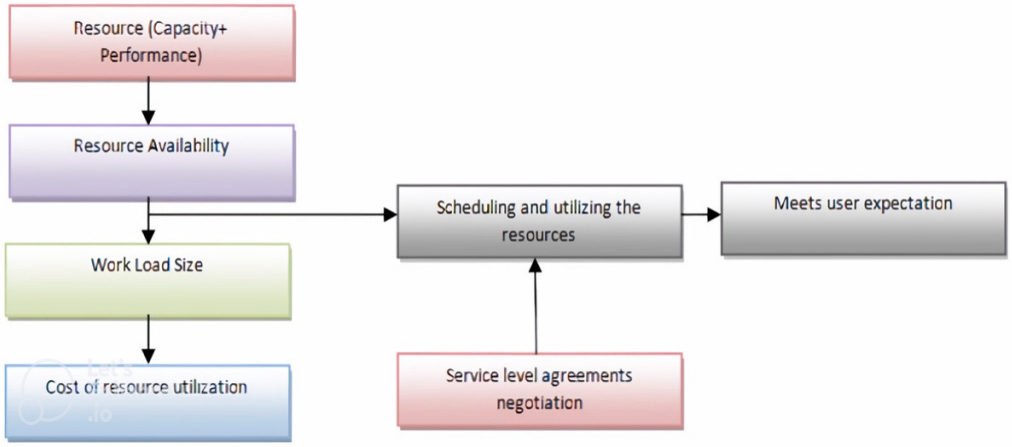

Figure 1. A framework for Resource Scheduling.

Time consumption is not only determined by hardware efficiency but also the efficiency of applications running on the system. The web and business applications may require high-performance computing machines to execute the application. The increasing cloud utility may demand the powerful machine in the cloud data center to perform high-performance computing. The increasing size of efficient and powerful cloud data centers consumes enormous amounts of cost and leads to high operational costs and carbon footprints.

In the view of attaining the timeliness of the computation, more patterns are employed to model a generic flow of work. The data parallelism pattern is appropriate to embarrassingly model a parallel computation of a data-intensive task. This pattern leads to the concurrent execution of multiple and independent data parallel tasks on heterogeneous computing of multi-core resources. However, data parallel task scheduling in heterogeneous environments with the aim of satisfying QoS constraints (such as cost and execution time) is a complex issue. The motivation of this work is the execution of data-intensive jobs within time and cost constraints using cloud heterogeneous resources. An efficient model is needed to acquire the performance of the cloud heterogeneous resources. The

data parallel task runs concurrently in cores that handle large amounts of data to often yield high throughput and performance. Therefore, task scheduling is presented here to implement task-level parallelism in cores. The contribution of this work is stated as follows. This paper proposes cost-optimized data parallel task scheduling in cloud resources. By running concurrent executions of data, the data parallel task is able to be finished within the deadline. This work experiments with a machine learning model that is replicated on the multi-core cloud heterogeneous resources to execute different input data concurrently for distributive learning. A linear mathematical model is developed here to reduce the operational cost of the data parallel task. The mathematical model aids in scheduling the load fractions to each multi-core resource or virtual machine feasibly. The experimental outcomes of the data-intensive task reveal that better solutions are attained in terms of deadline and keeping the computational cost at an optimized rate.

The organization of the paper is detailed as follows. Section 2 describes the related work. Section 3 discusses the need for data parallel task implementation on multi-core resources. The proposed time-constrained cost optimization model is formulated in Section 4. Experimental results and discussion are carried out in Section 5. Finally, Section 6 concludes the paper.

2. Related Works

Scheduling is essentially a decision-making mechanism that decides the execution order on the collection of available resources. The task scheduling on various computational environments, such as clusters, grids, and the cloud data center is crucial to complete execution within the deadline. The consumer submitting a job may have a deadline to comply with. If the target device is a power grid or a cloud, this information is useful, and the scheduler must use it to verify whether a given schedule complies with the constraints of the user. The performance of an application on heterogeneous systems is highly dependent on the processor's computing capacity, the number of processors, the bandwidth of communication, the size of the memory, etc. In [5], the author said that the appropriate scheduling mechanisms improve the Quality of Service (QoS) for cloud users by minimizing the total completion time of applications. The scheduling mechanisms aid in managing the complexity that is present in the management of distributed resources and allocations. The resources are scheduled based on the users' requirements such as the budget and deadline. The time cost-optimization scheduling algorithm aids in achieving a lower job completion time at a minimum computation cost. However, if more numbers of resources are used for computation, it makes the task more expensive to complete. Mixing data and task parallelism to compute the large computational applications often gives better speedups compared to either applying pure data parallelism or pure task parallelism.

In [6], the author proposed a proportional share allocation system that allows users to bid higher in order to gain more resource shares. Even if this allocation system intends to minimize the total run time and cost, it does not guarantee the completion of the workload within a fixed budget; moreover, troubles can be incurred due to the continuous intervention of the broker to prefer the cheapest resources to complete the task within the deadline [7]. The multiprocessor executes the parallel task simultaneously to obtain quick results, process a massive amount of data, and solve a problem in the expected time. Ref. [8] Machine learning technologies are applied to obtain acceptable solutions of tolerable time consumption for scheduling problems. Ref. [9] The workflow scheduling for efficient resource utilization of the volunteer computing system and cloud resources is applied. Therefore, many of the time-constrained parallel applications in a heterogeneous environment scheduled using efficient task-scheduling algorithms are completed. The existing task scheduling works of [10], Ref. [11] executed tasks on a core, but they did not implement parallelization on the core level. Good scheduling for a parallel task can make it meet its deadline. The existing scheduling works of [12–18] allocated all of the multiple available resources at a time to run the concurrency of tasks on the resources.

In [19], they stated that the deadlines could not be met by parallel tasks if parallel tasks are executed by one unique thread. Multi-core processors are capable of executing

the parallel task simultaneously to obtain quick results, process a massive amount of data, and solve a problem in the expected time [20]. This work uses multi-core resources to implement core-level parallelisms to obtain quick results and to process a massive amount of data within the expected time and minimum power consumption. Others can be seen in [21–29]. However, the complexity is increased by considering time and cost constraints. Due to the review above, the cost-optimized scheduling is formulated in this research to schedule the data parallel task in a multi-core environment under time and cost constraints.

3. Need of Data Parallel Task Implementation on Multi-Cores

The data parallel task is the simultaneous execution of the same task on different parts of the initial input dataset or task replication with different input datasets. The implementation of parallel execution requires the transfer of huge data subsets to remote virtual machines for concurrent execution and makes long wide-area data transfer latencies. They could be secured by the data management services through authorization and authentication of the remote site. The initiation of a new virtual machine creates latency and cost consumption. Hence, the completion of a task under the deadline and cost constraints is also complicated. Therefore, the probable number of parallel task executions in a minimum execution environment is needed, which is proposed here. It can be obtained by running concurrent executions on multi-core resources or virtual machines in the cloud. Thus, the resource utilization is optimized proficiently through executing the tasks concurrently on the multi-core resources.

If more executions are performed at the local sites, there is no need to transfer the data to the remote sites (i.e., data replicas in remote sites might be reduced, and security services can be achieved). In the cloud, there are k many-core machines or resources R with m cores. All the m cores in each machine or resource are identical. Here, the implementation of the data parallel task is performed through task replications, which run on different parts of the initial input dataset. The number of task replication depends on the number of cores; as a result, each task is run on an individual core. A multi-core virtual machine has more execution cores. Hence, multi-core resources are helpful for data parallel task scheduling to process a large amount of data concurrently to achieve high throughput and performance. Figure 2 indicates the scheduling of data parallel processing operations in a multi-core architecture. A multi-core processor improves the overall performance of the task by proper scheduling of parallel computation.

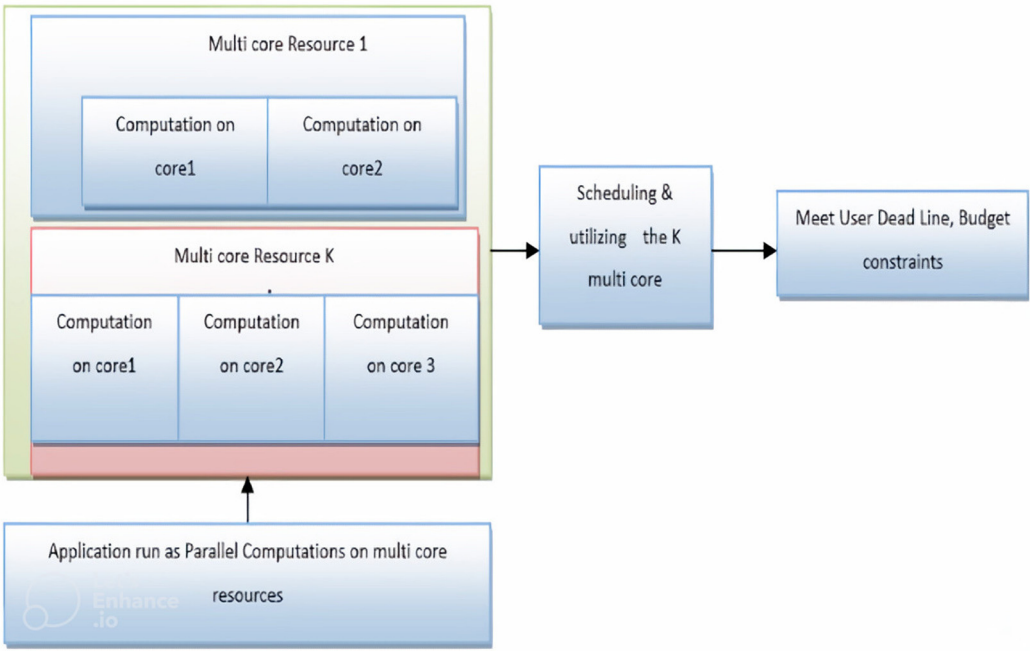

Figure 2. Concurrent computation on multi-core resources.

4. Scheduling Methodology

4.1. Problem Modeling

In this work, the proposed model consists of a finite set of k heterogeneous virtual machines or multi-core machines with varying capacities to execute a given task. These virtual machines can be represented as $R = \{R_1, R_2, R_3, \ldots, R_k\}$. The given data-intensive load N split into load fractions, which are $n_1, n_2, n_3 \ldots n_k$, are scheduled onto a set of multi-core machines $\{R_1, R_2, R_3 \ldots R_k\}$. The objective of the proposed work is described here:

- Replicate the task in order to increase the number of concurrent computations that depend on the number of cores on the virtual machine.
- Finding the feasible amount of data load fractions to achieve an optimized cost under a prefixed job deadline D and budget B constraints. While running a concurrent computation on the cores, each task runs independently on an isolated core without waiting for any hardware resources. The CPU time accounted for each task in CMP is approximately equal if identical workloads are running on a similar core.

The load fractions are communicated to the resources to run a replicated task concurrently on all the multi-core machines [30]. The execution of dependent tasks on the resources can make the overhead of communication which may dominate in the overall processing time. Hence, the overall turnaround time of load on a multi-core machine R_i will be calculated as the summation of the execution time and the communication time. In this work, it is assumed that there is no direct communication between virtual machines. The communication time (T_{di}) is related to the data transfer time, and the processing time (T_{ei}) is related to the execution time of load fractions n_i on the virtual machine R_i. Here, the execution time is linearly reliant on the fractions of load n_i *received by* the machine R_i. The communication time is linearly reliant on the fractions of load n_i transferred to the machine R_i. Moreover, the proposed model considers both communication and execution overhead on each machine R_i is raised during the task execution. Particularly, communication overheads are adapted to the latency on the network while transferring data to the sites. Similarly, execution overheads are due to scheduling the resource, task replication, etc. Here, $n_i \, \varepsilon \, N$. If n_i is the number of load fractions transferred into machine R_i, it is again partitioned into sub-load fractions such as n_{ci} to a set of cores on the same machine R_i.

$$n_{i\,=}\, n_{ci} * m_i \tag{1}$$

where m_i represents the numbers of cores on machine R_i, T_{ei} is the execution time of resource, which is given by

$$T_{ei} = e_i + t_{ei}\, n_{ci} \tag{2}$$

where e_i denotes the execution overheads, and the execution time of machine R_i is taken as the parallel execution time of cores on machine R_i. If the concurrent task execution on multiple cores runs with equal workload fractions n_{ci} on each core, the load fraction n_{ci} on a core is taken into consideration to calculate the execution time of virtual machine R_i. Here, t_{ei} is the proportionality constant between the time of execution and the number of load fractions n_{ci} submitted to the replicated task, which runs on the core of server R_i. Similarly, the communication time T_{di} is given by

$$T_{di} = d_i + t_{di} n_i \tag{3}$$

where d_i represents the communication start-up overheads, and t_{di} is the proportionality constant between the time of data transfer and the number of load fractions transferred to resource R_i. Accordingly, the total turnaround time T_i needed to execute n_i load fractions on R_i is calculated by

$$\begin{gathered} T_i = T_{ei} + T_{di} \\ T_i = T_{ei} + T_d = e_i + t_{ei}\, n_{ci} + d_i + t_{di}\, n_i \end{gathered} \tag{4}$$

4.2. Time and Cost-Constrained Cost Optimization

This section extends the model in Section 4.1 with time-constrained cost optimization. A cost optimization model is formed here, which distributes the load fractions to the selected multi-core resources or virtual machines to carry out the concurrent task in an optimal way. The following notations are used to formulate the mathematical model:

c_i—Cost required for concurrent processing of unit load task in resource R_i

n_{ci}—Part of the workload from sources distributed to the task that runs on the core of resource R_i

B—Budget for the given job.

t_{ei}—Concurrent execution time taken by executing a unit load on each core of resource R_i

t_{di}—Time taken to transfer a unit load to the resource R_i

N—Total workload of the given job.

m_i—The number of tasks that should be concurrently executed by virtual machine R_i

The objective is to minimize the total computational cost of a job under a prefixed job deadline D and cost constraints. The prefixed deadline and budget are given by the user. The total turnaround time of the job should be within the deadline.

$$Minimize = \sum_i c_i n_{ci} \tag{5}$$

$$\begin{aligned} e_i + t_{ei}\, n_{ci} + d_i + t_{di}\, m_i\, n_{ci} \leq \mathrm{D} \qquad & I = 1, \ldots \ldots k \\ t_{ei}\, n_{ci} + t_{di}\, m_i n_{ci} + o_i \leq \mathrm{D} \qquad & i = 1, \ldots \ldots k \end{aligned} \tag{6}$$

where $o_i = e_i + d_i$

$$\textstyle\sum_i c_i n_{ci} \leq B \;\; i = 1, k \tag{7}$$

$$\textstyle\sum_i m_i n_{ci} = \mathrm{N} \;\; i = 1, \ldots k \tag{8}$$

$$n_{ci} \geq 0. \tag{9}$$

Equation (5) represents the objective function. Equations (6)–(9) represent the constraints to attain the objective function. The constraints (6) and (7) represent the deadline and cost constraints of a given task. Constraint (8) represents that the load of the job is the summation of the load fractions, which are distributed for execution. Constraint (9) represents that the distributed workload fractions are non-negative constraints.

4.3. Time and Cost-Constrained Cost-Optimized Scheduling Algorithm

The resource manager examines the memory data and the performance of a virtual machine for running a unit of the workload, as well as computing the cost and communication time for jobs. The time and cost-constrained cost-optimized scheduling algorithm distribute the feasible load fractions to the selected multi-core resources based on a linear solution. Here, the replicated task is executed with different parts of submitted data loaded to the multi-core resources Algorithm 1.

Algorithm 1. Time and Cost-Constrained Cost-Optimized Scheduling Algorithm

Assumption: Resource manager should examine the value of t_{ei}, t_{di}, c_i, o_i for each resource
***Input*:** Task with input load N, user specified deadline D in sec and Budget B *in $.*
Variable Resource List [ ]
Variable Resource capacity [ ]//the values of unit load turnaround time t_i, c_i & m_i
***Output*:** load distribution based on the deadline and budget constrained cost optimization values & job is done at minimum cost from (5)–(9)

Begin
$i = 0$;
Get the load value of N;

Algorithm 1. Cont.

For each machine *R(i,k)* in Resource list[ ] do
If (*N* not null)
Get the value of c_i, Ti, mi of Virtual Machine R_i from Resource_Capacity [ ];
Get the feasible value n_{ci} of Machine R_i. based on the constraints of
$t_{ei}\ n_{ci} + t_{di}\ m_i\ n_{ci} + o_i \leq \mathrm{D}\ \&\ \sum_i \boldsymbol{c_i n_{ci}} \leq \mathrm{B}$
For resource R_i (j, m_i) do//where m_i is number of tasks can be executed by resource R_i
Create task replication tr_j on R_i
Distribute n_{ci} to task t_{rj} of machine R_i.
j++;
End
Execute all tasks concurrently;
Get Job Results.

4.4. Self-Learning Agent for Cost-Optimized Scheduling

A self-learning agent is designed here to automate the above cost-optimized scheduling of a task. The designed system is effective and interpretable for fitting the task into the machine. Based on the learning parameters from the environment, the system will schedule the job to an appropriate machine by considering the optimized cost and time constraints. In [31], the authors discussed the reinforcement learning agent that was adapted to the dynamic environment by learning the decision-making policy from experience. It is well appropriate for the resource management systems since the resource management scheduler needs to make scheduling decisions without knowing the dynamic change of job arrivals based on the machine's availabilities [32–34]. The decision agent repeatedly observes the learning parameters to make decisions to reach the optimal policy [35]. The Markov decision process is used for designing a self-learning agent. The agent perceives the environment state (s), action (a), and reward (r) from its learning experiences. The agent iteratively implements a state–action pair from its learning experiences to schedule the job for machines. The Markov chain has a stationary distribution Π and a finite mixing time T. A trajectory of samples of state actions s_1a_1, s_2a_2, s_3a_3, . . . , s_Ta_T has been chosen until its current state has a distribution that roughly matches Π. Here, an approximator function is used to adjust certain learning parameters to make a decision related to the optimal scheduling policy. The self-learning agent chooses the finest action based on the optimal selection policy π, which is given below.

$$\pi_{\theta}(a_t/s_t) = \min_{a \varepsilon A} L(a_t/s_t) \tag{10}$$

The agent is trained with different task machine pairs to accomplish the above scheduling policy. The selected trajectory from many action state pairs should map the task of the machines on the basis of minimized time and cost scheduling. The action $a_t \in A\ (s_t)$ and the state of the highest fitting of the machine job pair yields an immediate reward r from the environment. The value of L-learning has been updated from cumulative rewards r. α represents a transition of a state under action from the learning experience. The probability of transition is defined by

$$P_{\theta}(\alpha) = P\ (s_1a_1, \ldots\ \ldots\ , s_Ta_T) \tag{11}$$

$$P_{\theta}(\alpha) = \prod_{t=1}^{T} \boldsymbol{P}(\boldsymbol{s_t}/(\boldsymbol{s_{t-1}a_{t-1}})\boldsymbol{\pi\theta}(\boldsymbol{a_t}/\boldsymbol{s_t}) \tag{12}$$

The expectation of cumulative reward is defined as,

$$M(\theta) = E\alpha \sim [r\ (\alpha)] = \sum_{t=1}^{T} \boldsymbol{r}(s_t, a_t)\ P_{\theta}(\alpha) \tag{13}$$

$$r(\alpha) = r\ (s_1a_1, \ldots\ldots, s_Ta_T) = \sum_{t=1}^{T} \boldsymbol{r}_t = \sum_{t=1}^{T} \boldsymbol{r}(s_ta_t) \tag{14}$$

The gradient descent is applied to obtain cost-optimized scheduling:

$$\nabla_\theta \, E\alpha \sim [r(\alpha)] = \nabla_\theta \int r(\alpha) \, P_\theta(\alpha) \, d\alpha \quad (15)$$

The environment's state has a variable length of fitted multi-core machines and task pairs to achieve cost-optimized data parallel task scheduling. The agent will learn from its environment. The self-learning agent is trained up with different task and resource pairs to make an intelligent decision through iterative learning parameters from the environment. The main objective of learning is fitting the task to the machines under the constraints of minimum time and cost.

5. Results and Discussion

5.1. Experimental Evaluation

The cost-optimized data parallel task scheduling model is tested with resources that are listed in Table 1. A data-intensive task with an input size of 500,000 text data records under the constraints of a user-specified deadline of 5000 s, and a budget of 700 USD is taken here to evaluate the proposed work and algorithm. The algorithm uses the information of the resources mentioned in Table 1, in which the execution time t_{ei}(s) is calculated by concurrently running the unit workload on cores of virtual machine Ri. The time taken to transfer a unit load to the resource is t_{di}(s). The summation of the execution time and the communication time of the listed resources is calculated as t_i(s), which is given in Table 1. The summation of the execution, communication overheads, and concurrent processing cost of given resources are assumed and given in Table 1. For concurrent executions, those 500,000 records are distributed into the computing resources of multi-core virtual machines.

Table 1. Resource information.

Virtual Machine List	$o_{i\,(S)}$	t_i (s) per Unit Load ($t_{ei} + t_{di}$) (s)	Concurrent Processing Cost per Unit Load $c_{i\,(\$)}$	Optimal Load Fractions Value (Data Records)	Distributed Load Fractions to Each Server (Data Records)
R1	250	0.1364	0.00520	29,649	118,596
R2	250	0.1454	0.00260	32,668	130,672
R3	280	0.1498	0.00130	31,508	126,032
R4	280	0.1514	0.00065	31,175	62,350
R5	280	0.1514	0.00065	31,175	62,350

The following linear programming model is formed to give the feasible workload fraction values for each task or thread.

$$\text{MIN} = 0.00520 * n_1 + 0.00260 * n_2 + 0.00130 * n_3 + 0.00065 * n_4 + 0.00065 * n_5;$$

$$0.1364 * n_1 \leq 5000 - 250;$$

$$0.1454 * n_2 \leq 5000 - 250;$$

$$0.1498 * n_3 \leq 5000 - 280;$$

$$0.1514 * n_4 \leq 5000 - 280;$$

$$0.1514 * n_5 \leq 5000 - 280;$$

$$4 * n_1 + 4 * n_2 + 4 * n_3 + 2 * n_4 + 2 * n_5 = 500{,}000;$$

$$0.00520 * n_1 + 0.00260 * n_2 + 0.00130 * n_3 + 0.00065 * n_4 + 0.00065 * n_5 \leq 700;$$

The above linear equation has been solved. The cost-optimized scheduling gives the minimum processing cost of USD 320.5. The optimal load fraction values distributed to computing resources are shown in Table 1. The above resources have the capacity to execute the tasks concurrently. Hence, the task is replicated based on the number of cores in each virtual machine. Each of these is assigned with the optimal input of 29,649 records; thus 118,596 records are totally transferred to virtual machine R_1. The work is also tested up to schedule the task of different load values with different deadlines at the same budget of 700 USD. Figure 3 shows how the load fractions are distributed to each multi-core virtual resource to run the various load at optimal cost at different deadlines.

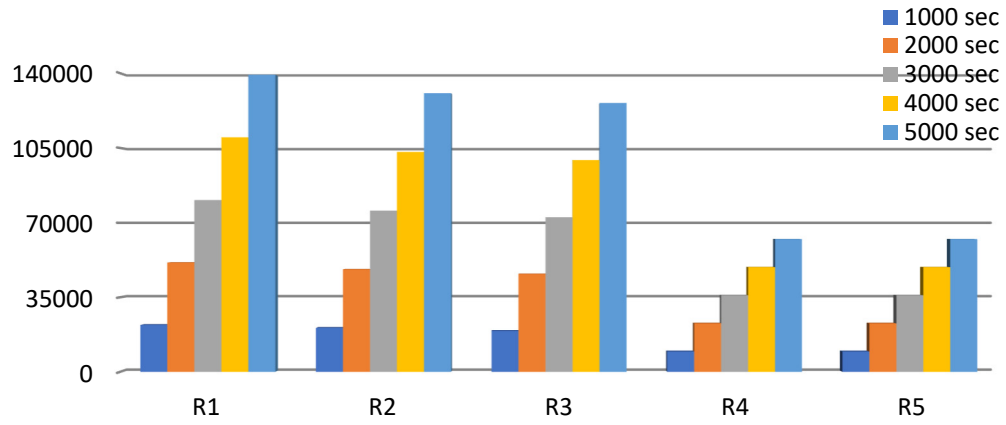

Figure 3. Task load processed by concurrent processes.

The experiment is performed on the assignment of the task to the multi-core virtual resources. In [36,37], the authors assigned the data-intensive job to a set of processors or clusters based on their speed and capacity to complete the task within the deadline. They considered the communication delay and data access latency to calculate the completion time of the task. The feasible assignment of the data load to each processor is based on processor capacity, hence the task assignment to virtual machines that have minimal cores is compared with the task assigned to a machine that has multiple cores.

The cost-optimized parallel task assignment experiments on available heterogeneous virtual resources that have minimal cores. However, it processes 317,236 records within the 5000 s deadline at the cost of 699.637 USD on five heterogeneous resources. It processes 500,000 records on five resources within 9000 s at the cost of 845.8 USD. As per our result, the concurrent executions on the multiple virtual machines that have minimal cores require more processing time to finish the submitted task load of 500,000 records; the execution of the same data parallel task on multi-cores of virtual machines is less within the deadline. In [38], the arrived task has a budget that limits the total number of resources because the running job has to pay the system for the usage of additional resources. Figure 4 shows the comparison of the execution on multi-core virtual resources and a virtual machine with minimal cores. Hence, the execution on a virtual machine with minimal cores under the constraints either requires more processing time or more resources.

The results show that the concurrent tasks that run the feasible load fractions on multi-core virtual machines are capable of processing the entire submitted load within the deadline at the minimum cost. However, the concurrent tasks on a virtual machine with minimal cores can process only around 60% of the given load within the deadline as shown in Figure 5. The cost-optimized data parallel task scheduling on multi-cores attains the optimized cost as shown in Figure 6. The highlight of this research considers task replication in the core processors to minimize the time and cost. If the multi-core virtual resources are available, the task could be executed as per data parallel task scheduling within the minimum span; otherwise, examining the currently available resources may increase the execution time and cost.

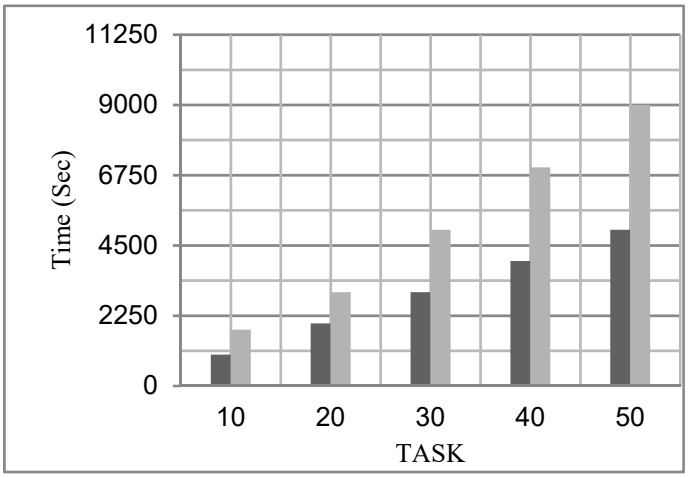

Figure 4. Execution Time of Data-Intensive Tasks.

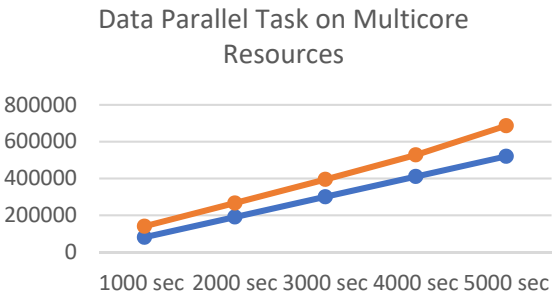

Figure 5. Data parallel task on multi-core resources.

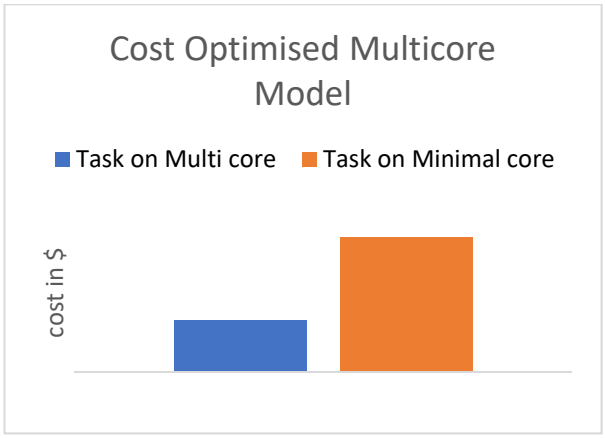

Figure 6. Cost-optimized multi-core model with a deadline of 5000 s.

5.2. Experimental Evaluation for Data Parallelism Using Model Replication

Nowadays, machine learning and deep learning play vital roles in all multi-disciplinary applications [39,40]. The models are developed to predict or classify the image data or text data. The models are developed and deployed for prediction, classification, and recommendation. While developing the models, training and testing processes are carried out. A total of 70% of text data or image data will be used for training, and 30% of text data or image data will be used for the testing process to improve the learning rate of the machine. If the dataset size is large, a huge number of image or text data could be used for training and testing process. It will consume time and money to train and test the model. The cost-optimized data parallel task scheduling is applied to train and test the model. The highlight of this research considers distributive learning by applying machine

learning model replication in the core processors to make the time and cost minimam. The data parallel task is the simultaneous execution of the same model on different parts of the input image dataset or the machine learning model being replicated to execute different input datasets concurrently. Hence, an image data-intensive task of the classification of leaf images is experimented with here. The input leaf image is shown in Figure 7; there is a data size of 100,000 images taken here to train and test the model.

Figure 7. Sample Augmented Images.

The algorithm uses the information about virtual resources. The primary idea behind data parallelism is to boost the overall sample throughput rate by duplicating the model over multiple machines and performing backpropagation in parallel to obtain more information about the loss function faster. Finally, the various results are combined and merged to create a new model [41,42]. The execution time t_{ei} is calculated based on the concurrent processing time of an image on each core of virtual machine R_i. Therefore, the machine learning model is replicated on each core of resource R_i. The time taken to transfer a unit load to the resource is signified as t_{di}. The summation of the execution time and the communication time of resources are considered for the data parallel model implementation. Distributed learning gives better performance instead of running the model in a machine, since batchwise running is executed to implement distributed learning.

The distributed training of deep learning models is implemented on both multi-core machines and multiple machines or clusters. The multi-core machine gives better throughput compared with multiple machines, as shown in Figure 8.

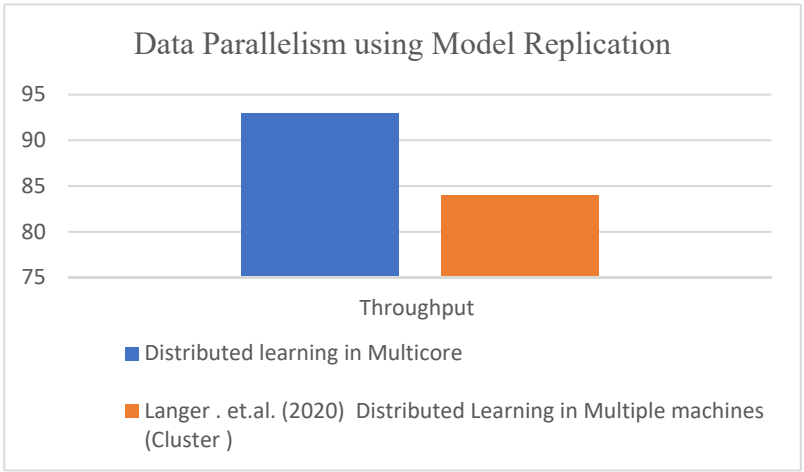

Figure 8. Comparison of execution of data parallel task on multiple machines with multi-cores [41].

6. Conclusions and Future Work

In this paper, we proposed an improvement to data parallel task-scheduling by allocating cores to the tasks with the intention of minimizing the overall execution time and cost margin. The machine learning application with a bulky dataset was deployed to examine

the effectiveness of the concurrent execution on multi-core resources. A cost-optimized data parallel task scheduling on multi-core resources is formulated. The experimental results reveal that the concurrent execution of tasks in a multi-core environment is able to complete the application within the deadline. The optimal solutions are attained in terms of processing the data parallel task by the deadline while keeping the computational cost at an optimized rate. Further studies will investigate a specific application that implements the proposed algorithm. The specific application is the comparison of machine learning models that will be performed in parallel through the proposed task and data parallelism scheduling process within the time and cost constraints.

Author Contributions: Conceptualization, N.R.R., A.D. and M.K.; methodology, R.S., A.G. and S.V.A.; validation, R.S., A.G., S.V.A. and D.A.; formal analysis, R.S., A.G., S.V.A., D.A., I.D.N. and D.H.E.; investigation, I.D.N. and D.H.E.; resources, R.S. and A.G.; data curation, R.S., A.G., S.V.A. and D.A.; writing—original draft preparation, N.R.R., A.D. and M.K. All authors have read and agreed to the published version of the manuscript.

Funding: The research was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R238), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R238), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hajikano, K.; Kanemitsu, H.; Kim, M.W.; Kim, H.-D. A Task Scheduling Method after Clustering for Data Intensive Jobs in Heterogeneous Distributed Systems. *J. Comput. Sci. Eng.* **2016**, *10*, 9–20. [CrossRef]
2. Kezia Rani. B, Vinaya Babu, A. Scheduling of Big Data Application Workflows in Cloud and Inter-Cloud Environments. In Proceedings of the 2015 IEEE International Conference on Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 78–84.
3. Arunarani, A.R.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2018**, *91*, 407–415. [CrossRef]
4. Cheng, D.; Rao, J.; Jiang, C.; Zhou, X. Resource and Deadline-aware Job Scheduling in Dynamic Hadoop Clusters. In Proceedings of the IEEE 29th International Parallel and Distributed Processing Symposium, Hyderabad, India, 25–29 May 2015.
5. Khan, T.; Singh, K.; Son, L.H.; Abdel-Basset, M.; Long, H.V.; Singh, S.P.; Manjul, M. A Novel and Comprehensive Trust Estimation Clustering Based Approach for Large Scale Wireless Sensor Networks. *IEEE Access* **2019**, *7*, 58221–58240. [CrossRef]
6. Sahu, D.P.; Singh, K.; Manju, M.; Taniar, D.; Abdel-Basset, M.; Long, H.V. Heuristic Search Based Localization in Mobile Computational Grid. *IEEE Access* **2019**, *7*, 78652–78664. [CrossRef]
7. Kaur, G. A DAG based Task Scheduling Algorithms for Multiprocessor System—A Survey. *Int. J. Grid Distrib. Comput.* **2016**, *9*, 103–114. [CrossRef]
8. Xu, H.; Liu, Y.; Lau, W.C.; Guo, J.; Liu, A. Efficient Online Resource Allocation in Heterogeneous Clusters with Machine Variability. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May2019; pp. 89–95.
9. Ghafarian, T.; Javadi, B. Cloud-aware data intensive workflow scheduling on volunteer computing systems. *Futur. Gener. Comput. Syst.* **2014**, *51*, 87–97. [CrossRef]
10. Thoman, P.; Dichev, K.; Heller, T.; Iakymchuk, R.; Aguilar, X.; Hasanov, K.; Nikolopoulos, D.S. A taxonomy of task-based parallel programming technologies for high-performance computing. *J. Supercomput.* **2018**, *74*, 1422–1434. [CrossRef]
11. Tyagi, R.; Gupta, S.K. A Survey on Scheduling Algorithms for Parallel and Distributed Systems. In *Silicon Photonics & High Performance Computing; Advances in Intelligent Systems and Computing*; Springer: Singapore, 2018; Volume 718. [CrossRef]
12. Bharadwaj, V.; Ghose, D.; Robertazzi, T.G. Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. *Clust. Comput.* **2003**, *6*, 7–17. [CrossRef]
13. Buyya, R.; Murshed, M. *A Deadline and Budget Constrained Cost-Time Optimisation Algorithm for Scheduling Task Farming Applications on Global Grids*; Technical Report CSSE-2002/109; Monash University: Melbourne, Australia, 2002.

14. Celaya, J.; Arronategui, U. Fair scheduling of bag-of-tasks applications on large-scale platforms. *Futur. Gener. Comput. Syst.* **2015**, *49*, 28–44. [CrossRef]
15. Gómez-Martín, C.; Vega-Rodríguez, M.A.; González-Sánchez, J.L. Fattened backfilling: An improved strategy for job scheduling in parallel systems. *J. Parallel Distrib. Comput.* **2016**, *97*, 69–77. [CrossRef]
16. Mishra, S.K.; Sahoo, B.; Parida, P.P. Load balancing in cloud computing: A big picture. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *32*, 149–158. [CrossRef]
17. Priya, B.; Gnanasekaran, T. To optimize load of hybrid P2P cloud data-center using efficient load optimization and resource minimization algorithm. *Peer-to-Peer Netw. Appl.* **2019**, *13*, 717–728. [CrossRef]
18. Raja, C.V.; Jayasimman, D.L. A Cost Effective Scalable Scheme for Dynamic Data Service in Heterogeneous Cloud Environment. *Int. J. Adv. Sci. Technol.* **2019**, *28*, 764–776.
19. Kuo, C.-F.; Lu, Y.-F. Scheduling algorithm for parallel real-time tasks on multiprocessor systems. *ACM SIGAPP Appl. Comput. Rev.* **2016**, *16*, 14–24. [CrossRef]
20. Luque, C.; Moreto, M.; Cazorla, F.J.; Gioiosa, R.; Buyuktosunoglu, A.; Valero, M. CPU Accounting for Multicore Processors. *IEEE Trans. Comput.* **2012**, *61*, 251–264. [CrossRef]
21. Alebrahim, S.; Ahmad, I. Task scheduling for heterogeneous computing systems. *J. Supercomput.* **2017**, *73*, 2313–2338. [CrossRef]
22. Blake, G.; Dreslinski, R.G.; Mudge, T. A Survey of multi-core processors. *IEEE Signal Process. Soc.* **2009**, *26*, 45–53.
23. Chen, Y.; Alspaugh, S.; Borthakur, D.; Katz, R. Energy efficiency for large-scale MapReduce workloads with significant interactive analysis. In Proceedings of the 7th ACM European Conference on Computer Systems, Bern, Switzerland, 10–13 April 2012; pp. 43–56. [CrossRef]
24. Ghafouri, R.; Movaghar, A.; Mohsenzadeh, M. A budget constrained scheduling algorithm for executing workflow application in infrastructure as a service clouds. *Peer-to-Peer Netw. Appl.* **2018**, *12*, 241–268. [CrossRef]
25. Khan, M.A. Task scheduling for heterogeneous systems using an incremental approach. *J. Supercomput.* **2017**, *73*, 1905–1928. [CrossRef]
26. Popa, A.; Hnatiuc, M.; Paun, M.; Geman, O.; Hemanth, D.J.; Dorcea, D.; Ghita, S. An Intelligent IoT-Based Food Quality Monitoring Approach Using Low-Cost Sensors. *Symmetry* **2019**, *11*, 374. [CrossRef]
27. Sulistio, A.; Buyya, R. A Time Optimization Algorithm for Scheduling Bag-of-Task Applications in Auction-based Proportional Share Systems. In Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05), Rio de Janeiro, Brazil, 24–27 October 2005. [CrossRef]
28. Terzopoulos, G.; Karatza, H.D. Power-aware Bag-of-Tasks scheduling on heterogeneous platforms. *Clust. Comput.* **2016**, *19*, 615–631. [CrossRef]
29. Pop, F.; Dobre, C.; Cristea, V.; Bessis, N.; Xhafa, F.; Barolli, L. Deadline scheduling for aperiodic tasks in inter-Cloud environments: A new approach to resource management. *J. Supercomput.* **2015**, *71*, 1754–1765. [CrossRef]
30. Rho, J.; Azumi, T.; Nakagawa, M.; Sato, K.; Nishio, N. Scheduling Parallel and Distributed Processing for Automotive Data Stream Management System. *J. Parallel Distrib. Comput. (JPDC)* **2017**, *109*, 286–300. [CrossRef]
31. Rajalakshmi, N.R.; Arulkumaran, G.; Santhosh, J. Virtual Machine Consolidation for Performance and Energy Efficient Cloud Data Centre using Reinforcement Learning. *Int. J. Eng. Adv. Technol.* **2019**, *8*, 78–85.
32. Ranaldo, N.; Zimeo, E. Time and Cost-Driven Scheduling of Data Parallel Tasks in Grid Workflows. *IEEE Syst. J.* **2009**, *3*, 104–120. [CrossRef]
33. Yang, J.; He, Q. Scheduling Parallel Computations by Work Stealing: A Survey. *Int. J. Parallel Program.* **2018**, *46*, 173–197. [CrossRef]
34. Marri, N.P.; Rajalakshmi, N. MOEAGAC: An energy aware model with genetic algorithm for efficient scheduling in cloud computing. *Int. J. Intell. Comput. Cybern.* **2021**. [CrossRef]
35. Rajalakmi, N.R.; Balaji, N. A Vikor Method For Distributing Load Balanced Virtual Machine in Cloud Data Center. *Int. J. Appl. Eng. Res.* **2015**, *10*, 10127–10136.
36. Xiong, F.; Yeliang, C.; Lipeng, Z.; Bin, H.; Song, D.; Dong, W. Deadline based scheduling for data-intensive applications in clouds. *J. China Univ. Posts Telecommun.* **2016**, *23*, 8–15. [CrossRef]
37. Wang, B.; Song, Y.; Sun, Y.; Liu, J. Managing Deadline-constrained Bag-of-Tasks Jobs on Hybrid Clouds with Closest Deadline First Scheduling. *KSII Trans. Internet Inf. Syst.* **2016**, *10*, 2952–2971. [CrossRef]
38. Tripathy, B.; Dash, S.; Padhy, S.K. Dynamic task scheduling using a directed neural network. *J. Parallel Distrib. Comput.* **2015**, *75*, 101–106. [CrossRef]
39. Zhang, S. Distributed Stochastic Optimization for Deep Learning. Ph.D. Dissertation, New York University, New York, NY, USA, 2016.
40. Zheng, S.; Meng, Q.; Wang, T.; Chen, W.; Yu, N.; Ma, Z.M.; Liu, T.Y. Asynchronous Stochastic Gradient Descent with Delay Compensation. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 4120–4129.
41. Langer, M.; He, Z.; Rahayu, W.; Xue, Y. Distributed Training of Deep Learning Models: A Taxonomic Perspective. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 2802–2818. [CrossRef]
42. Roy, S.K.; De, D. Genetic Algorithm based Internet of Precision Agricultural Things (IopaT) for Agriculture 4.0. *Internet Things* **2020**, *18*, 100201. [CrossRef]