Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 07, No. 04, December 2021.

WORKFLOW SCHEDULING ACCORDING TO DATA DEPENDENCIES IN COMPUTATIONAL CLOUDS

Hamid Saadatfar[1] and Batoul Khazaie[2]

(Received: 19-Jul.-2021, Revised: 19-Sep.-2021, Accepted: 5-Oct.-2021)

ABSTRACT

The number of applications needing big data is on the rise nowadays, where big data processing tasks are sent as workflows to cloud computing systems. Considering the recent advances in the Internet technology, cloud computing has become the most popular computing technology. The scheduling approach in cloud computing environments has always been a topic of interest to many researchers. This paper proposes a new scheduling algorithm for data-intensive workflows based on data dependencies in computational clouds. The proposed algorithm tries to minimize the makespan by considering the details of the workflow structure and virtual machines. The concepts and details defined and considered in this study have received less emphasis in previous works. According to the results, the proposed algorithm reduced the duration of communication between tasks and runtimes by taking into account the features of data-intensive workflows and proper task assignment. Consequently, it reduced the total makespan in comparison with previous algorithms.

KEYWORDS

Bottleneck task, Computational volume, Data-intensive applications, Resource allocation, Sensitive task, Workflow scheduling.

1. INTRODUCTION

The current computing structures for today's applications are growing in the form of a variety of networks, clusters and clouds, among which cloud computing has rapidly become a widely acceptable sample for scientific experiments, big data analysis and data-intensive applications. In such a distributed computing environment, data-intensive applications require high-performance computing resources to facilitate proper execution of tasks [1][2]. Scheduling is a very important problem in cloud computing systems, a goal of which is to make optimal and efficient use of resources and respond to users in real time. Due to the massive system scale and inherent complexity of applications, workflow scheduling is considered an NP-hard problem. Most of the big data applications contain hundreds of closely related tasks requiring to read or write massive amounts of data [3]. These applications which usually need to interact with their data are called data-intensive applications. The makespan of a data-intensive application is a very important efficiency criterion, which can be affected by many factors, such as the task scheduling mechanism, server load, communications and data access delay. For instance, the network performance can significantly affect data access delay and, as a result, the makespan [4][5][6]. The Directed Acyclic Graph (DAG) model is a popular, effective and common method of displaying complicated applications [7][8][9][10]. The workflows in data-intensive applications usually act as a series of interconnected tasks with data/control dependencies [11]. Increasing communication costs is evidently ideal for performance enhancement, especially for data-intensive applications. However, this is not always possible for various reasons. For example, data might be located in the local storage facilities, but the user is forced to outsource the computation to a cloud because of overloaded local processing nodes or needs to higher computing capacity [12]. Since the workflow scheduling of data-intensive applications has been considered a serious problem recently, this paper proposes a low-complexity heuristic algorithm to improve the runtime by considering data exchange, workload balance, properties of data-intensive applications and heterogeneity of machines.

The proposed algorithm attempts to reduce the average makespan by considering the properties of applications needing big data by defining new concepts, such as sensitive tasks and bottleneck tasks and scheduling based on considering machines' characteristics (such as powerful or weak

1. H. Saadatfar (corresponding author) is with Department of Computer Engineering, University of Birjand, Birjand, Iran. Email: saadatfar@birjand.ac.ir, ORCID: 0000-0002-6130-8450.
2. B. Khazaie is with Department of Computer Engineering, Islamic Azad University, Birjand, Iran. Email: b.khazaie63@Gmail.com

communicating capabilities), noting the homogeneity and heterogeneity of tasks in the workflows of data-intensive applications and resources in the cloud environment. The proposed algorithm analyzes the workflow's graph structure and identifies the important subtasks (sensitive and bottleneck subtasks) considering the degree of vertices and the number of parents and children to reduce the runtime by striking balance in task mapping between machines. The algorithm consists of two steps to find the best machine for task assignment. The first step identifies and assigns sensitive tasks and the second step assigns insensitive and bottleneck tasks.

The rest of this paper is structured as follows: Section 2 reviews the literature. Section 3 presents the relevant concepts. In Section 4, the proposed algorithm and solution are introduced. Section 5 presents the simulation results. Finally, Section 6 reaches a conclusion and discusses future works.

2. RELATED WORKS

Cloud computing environments are nowadays used in nearly every field, such as engineering, mathematics, fundamental sciences and biotechnology. Significant studies have been conducted on management, scheduling and resource allocation and provisioning in cloud computing environments [3][7][13]. Workflow scheduling is used as an effective tool for system efficiency and performance enhancement. The workflows of data-intensive tasks (refer to large-scale data) are increasingly becoming more common in today's applications. Thus, scheduling algorithms should be modified and redesigned with respect to the specific features of these data-intensive workflows. Some of the studies on scheduling and resource allocation are reviewed in this section.

The paper presented by [14] introduced a workflow scheduling method for large-scale distributed systems from the perspective of Quality of Service (QoS) and data location analysis. The goal of this method is usually to provide a single relation and a scalable storage solution for cloud applications through storage on public services. The paper presented by [15] proposed an efficient algorithm for cloud workflow scheduling named Efficient Workflow Scheduling Algorithm (EWSA), which can manage a large number of applications simultaneously. The goal of this algorithm is to estimate the runtimes of all dynamic resources in order to maximize the use of resources and execute workflows in predetermined intervals. The paper presented by [16] introduced a Grouped Task Scheduling (GTS) algorithm by using QoS to estimate user requirements. This scheduler employs the min-min algorithm for the prioritization of batches. The authors in [17] employed the division algorithm for cloud computing scheduling with various databanks. A Divisible Load Theory (DLT) scheduling strategy is used for data-intensive computational loads in a heterogeneous cloud computing environment to minimize the total makespan and maximize the system efficiency by recovering a partitioned load from numerous databanks with respect to the distribution rate of databanks and the speed of each role. The paper presented by [18] analyzed task scheduling in the cloud environment by adopting a soft computing technique, in which the Genetic Algorithm (GA) was integrated with fuzzy sets for load balancing in the cloud environment. The final goal was to reduce the makespan in the cloud environment, so that computing resources would not be wasted. The paper presented by [19] introduced a Modified Genetic Algorithm (MGA) for resource-scheduling and proposed a workflow framework for cloud computing with a resource scheduling mechanism to improve the use of resources and system efficiency with respect to the QoS requirements.

The paper presented by [20] proposed an algorithm based on the Augmented Shuffled Frog Leaping Algorithm (ASFLA). The proposed algorithm determines the number of available virtual machines and employs the ASFLA to map the tasks in batches onto virtual machines for makespans. The paper presented by [21] introduced a workflow-scheduling technique aware of the data transfer duration and location based on the network bandwidth in a distributed environment with heterogeneous resources in addition to a local resource management method for data workflows in the cloud environment. The proposed algorithm employs the data workflow parallelization technique along the execution of tasks to reduce the execution cost of a workflow. In the paper presented by [22], a data-intensive application workflow-scheduling method was proposed for the heterogeneous computing environment (I-PDEA: Improved Partition-based Data-intensive Workflow Optimization Algorithm) to enhance the system throughput by

partitioning data workflows and mapping each partition to the available heterogeneous resources having the minimum runtimes. A Partial Critical Path (PCP) algorithm was introduced in [11] to minimize the workflow execution costs while satisfying the makespan constraints. The Multi-Cloud Partial Critical Paths with Pre-treatment (MCPCPP) algorithm reduces the execution cost of a workflow in a makespan by finding multiple critical paths in that workflow and assigning them to the available computing services with the lowest cost. The paper presented by [23] proposed a scheduling algorithm, named the Granularity Score Scheduling (GSS) based on the details of the tasks in a workflow to minimize the runtime and maximize the efficiency of the workflow system in data-intensive applications. The proposed algorithm consists of three phases named level B (the maximum task length of an output task), task ranking and task mapping onto virtual machines. The paper presented by [24] introduced a scheduling algorithm which not only considered the computing capacity of existing virtual machines, but also the connection and access delays. The proposed algorithm also considered the different makespans of tasks to reduce the runtime by allocating tasks to more powerful machines. In addition, some of the subtasks can be executed simultaneously.

The paper presented by [25] introduced a scheduling algorithm on the volunteer computing systems. For this purpose, workflows were divided into sub-workflows to minimize data dependencies. The sub-workflows were then assigned to the distributed voluntary resources (executive nodes) with respect to the proximity of resources and load distribution policies. If the sub-workflows miss their sub-makespans due to a long waiting time, the scheduling will take place on the public cloud resources. The paper presented by [26] proposed a mechanism based on the workflow structure to identify the number of necessary virtual machines, configure these machines based on the aforementioned structure and optimize data transfer between tasks. This algorithm schedules tasks in a way that the number of executed tasks in each sample equals the number of virtual machines employed to run the workflow. The authors in [27] introduced a QoS-based workflow scheduling method and described the minimum effect imposed by the new input data to force rescheduling the workflow process. For this purpose, a database was used for data storage. The proposed algorithm is run through the Markov chain by scheduling the most complicated branch to the least complicated branch and allocating them on highly accessible machines with low costs. Given the importance of workflow scheduling in data-intensive applications and relevant studies conducted in recent years, most of the proposed approaches have focused on optimization of general scheduling and runtime minimization. None of the studies dealt with the different features of machines and the effects they might have on the execution of the DAG workflow. The different structures of machines have also been neglected.

The scheduling approach proposed in [28] considers the impact of communication between two tasks when building the schedule of a scientific workflow. It attempts to assign pairs of tasks with significant data transfers to the same computational node in order to minimize the overall communication cost. The authors in [29] proposed an immune particle swarm optimization algorithm (IMPSO) to solve the workflow scheduling problem with more speed and quality. In solving this problem, they have considered optimizing both execution time and cost criteria. In [30], the authors proposed a multi-objective workflow scheduler based on a prediction-based dynamic evolutionary algorithm. They also employed neural network to improve the answers' quality. They considered resources' failures to achieve more reliable task-resource mappings. They considered the estimated time to transfer data as a parameter in the objective function.

Researchers in [31] proposed a list-based scheduling algorithm called CAS-L1, which is a contention- aware algorithm. CAS-L1 is a heuristic scheduler based on lookahead technique, which schedules data transfers explicitly. Although this scheduler is a successful one for data-intensive workflows, it does not take into account the data exchanges of all workflows running at the same time and even other network traffic. The scheduler proposed in [32] called Minimum Communication Cost (MCC) algorithm focused on links' available bandwidth and data files' size that should be transformed. It can minimize communications effectively; however, lack of attention to the computing power of machines and their different ability to run different tasks is a point that has received less attention in this algorithm.

Compared to these previous works, we have defined and considered new concepts to focus on the data dependencies of a workflow that help map sub-tasks to a more appropriate machine. Also,

the distinct ability of machines to perform different tasks has been considered in achieving better mappings. This paper tried to propose a novel approach by defining new concepts, considering different capabilities of machines to execute data-intensive tasks with respect to memory, processing speed, storage space, relevant effects on task execution, different structures of data-intensive workflows and complying with balance in existing machines and necessary data.

3. RELEVANT CONCEPTS

3.1 Sensitive Task (T_S)

If the runtime of a task on some machines differs significantly from that on other machines, this task is called sensitive. In fact, when the runtimes of a task on different machines are sorted in an ascending order, the task is defined as sensitive if its runtime on one machine is at least two times longer than its runtime on the previous one [33]. This new concept has been defined in order to identify tasks the execution time of which has undergone many changes according to existing computing resources and to allocate resources with more attention to them.

3.2 Competent Machine

A machine or a series of machines can be considered competent to execute a task if the task's runtime on them is significantly shorter (at least a half) than the runtime on other machines. For instance, a machine with sufficient memory or high processing power is the competent machine for a task if it is capable of executing that task at a significantly shorter time interval compared to the other machines. Therefore, a competent machine is determined according to a specific task and the set of competent machines for a task includes all the machines from the whole cloud computing system that have a special ability to run that program and can do it at a run time at least a half of the run time on other machines.

Depending on the characteristics of the tasks and the computing power of the resources, the execution time of the tasks will vary on different computing machines. The more the capabilities of a resource are in line with the requirements of a task, the more execution time will be reduced. The concept of a competent machine is defined so that the same fact (the suitability of some resources to perform certain tasks) can be considered in the scheduling problem.

3.3 Bottleneck Task (Tbn)

A bottleneck task in a workflow is defined to be the one which has a considerably high data dependency on the other tasks of the workflow. Considering the graph structure of a workflow, a node is called a bottleneck if its degree is three times higher than the average degree of the total nodes in that graph. The existence of sub-tasks that create bottlenecks in the execution of the workflow is common in parallel and distributed processing and therefore, this new definition can help identify these tasks and allocate a better resource to them.

3.4 Machines with Fast and Slow Connections

A machine is called fast-connected if the average delay time of its directly-connected channels is fewer than a half of the average delay time of all other machines. However, if this delay time is two times higher compared to the other machines, it is called slowly-connected. Paying attention to resource characteristics can help better map tasks. Recognizing high-speed communication machines will provide good options for bottleneck tasks.

3.5 The Share of Each Virtual Machine from Computations

The computation share of a virtual machine from executing a workflow is calculated as follows:

$$CV = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} \frac{T_{ij}}{m}}{m} \quad (1)$$

where *m* and *n* are parameters that indicate the number of machines and tasks, respectively and

T_{ij} represents the execution time of task j on machine i. Since a balanced division of tasks between machines can help increase efficiency, calculating the estimated share of each virtual machine in total workflow processing can be a good indicator of achieving this balance and determining the appropriate amount of load to be placed on each resource.

4. THE PROPOSED ALGORITHM

4.1 Preliminary Definitions

A data-intensive workflow is represented by a directed acyclic graph (DAG). Each workflow consists of edges showing the dependency between tasks. A time limit is defined as a deadline for each workflow. In this case, a workflow like w contains n tasks $\{T_1, T_2, \ldots, T_n\}$. The system model includes m virtual machines $\{Vm_1, Vm_2, \ldots, Vm_m\}$ and E shows the edges in the workflow graph. Figure 1 indicates a data-intensive workflow. The edge between i and j means that task T_i is the prerequisite of task T_j. In other words, T_j can start only after T_i is completed. The weights of edges indicate the time relationship between every two tasks.

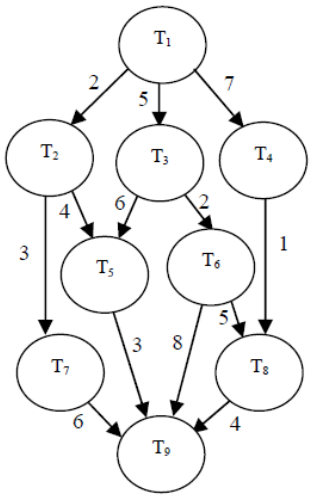


Figure 1. A DAG for a data-intensive application in the cloud computing environmen.

The makespan is the total runtime of all tasks in a workflow and it is defined as follows:

$$Ms=Mmax(Ms(VMj)) \tag{2}$$

Expected Time to Compute (ETC_{ij}) is a matrix with i rows and j columns ($1< i< n$, $1< j< m$) which shows the runtimes of tasks on different machines:

$$ETC = \begin{matrix} \\ T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{n-1} \\ T_n \end{matrix} \begin{matrix} \begin{matrix} Vm_1 & Vm_2 & Vm_3 & & Vm_{m-1} \end{matrix} \\ \begin{bmatrix} ETC_{11} & ETC_{12} & ETC_{13} & \cdots & ETC_{1m} \\ ETC_{21} & ETC_{22} & ETC_{23} & & ETC_{2m} \\ ETC_{31} & ETC_{32} & ETC_{33} & & ETC_{3m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ETC_{(n-1)1} & ETC_{(n-1)2} & ETC_{(n-1)3} & \cdots & ETC_{(n-1)m} \\ ETC_{n1} & ETC_{n2} & ETC_{n3} & & ETC_{mn} \end{bmatrix} \end{matrix} \tag{3}$$

Moreover, CT_{ij} ($1 < i, j< m$) is introduced as a matrix indicating the relationship between VM_i and VM_j based on the communication times:

$$CT = \begin{matrix} \\ Vm_1 \\ Vm_2 \\ Vm_3 \\ \vdots \\ Vm_{m-1} \\ Vm_m \end{matrix} \begin{matrix} \begin{matrix} Vm_1 & Vm_2 & Vm_3 & & Vm_{m-1} & Vm_m \end{matrix} \\ \begin{bmatrix} 0 & CT_{12} & CT_{13} & \cdots & CT_{1(m-1)} & CT_{1m} \\ CT_{21} & 0 & CT_{23} & & CT_{2(m-1)} & CT_{2m} \\ CT_{31} & CT_{32} & 0 & & CT_{3(m-1)} & CT_{3m} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ CT_{(m-1)1} & CT_{(m-1)2} & CT_{(m-1)3} & & 0 & CT_{(m-1)m} \\ CT_{m1} & CT_{m2} & CT_{m3} & \cdots & CT_{m(m-1)} & 0 \end{bmatrix} \end{matrix} \tag{4}$$

As discussed earlier, the goal of scheduling in cloud computing is to reduce the runtime by assigning the tasks of a data-intensive application to machines such that the total makespan is minimized.

4.2 The Proposed Scheduling Algorithm

The proposed algorithm is a static workflow scheduling algorithm for the cloud computing environment. It is applied to the applications which require big data. It consists of two phases: sensitive tasks' scheduling and insensitive/bottleneck tasks' scheduling. The following pseudocode shows the details of the process.

Algorithm 1: Pseudocode of the Proposed Algorithm

Inputs: A DAG D = (T, E), an ETC matrix of size n × m and a CT matrix of size n × n
Output: Schedule S of DAG

```
1  Tsort =Sort ETC Matrix Rows in Ascending Order
2  sensitiveTasks=[ ]
3  BestMachines=[ ]
4  for i=1 to n
5    for j=2 to m
6      if Tsort(i,j)/Tsort(i,j-1) >=2
7        Add i To sensitiveTasks set   // finding sensitive tasks
8        BestMachines(i)=[M1, M2, ..., Mj]   //since the runtimes are sorted ascending
9      end if
10   End for j
11 End for i
12 AvgDeg=Calculate the average degrees of vertices of the DAG D
13 BottleNodes=[ ]
14 for i=1 to n
15   if  degrees of vertice(i) >= 3* AvgDeg
16     Add i To BottleNodes set   // finding bottleneck tasks
17   end if
18 end for i
19 AvgCommunications = Calculate the total average communication time based on CT matrix
20 FastConnectionMachines=[ ]
21 SlowConnectionMachines=[ ]
22 For i=1 to n
23   if  average time of communication for machine(i) <= AvgCommunications/2 then
24     Add i To FastConnectionMachines set   // finding fast connected machines
25   end if
26   if  average time of communication for machine(i) >= AvgCommunication*2 then
27     Add i To SlowConnectionMachines set
28   end if
29 end for i
30 Sort sensitiveTasks by runtime in Descending Order
31 sensitiveTasks = sensitiveTasks - BottleNodes
32 ComputationalVolume = Sum(Average(runtime Tasks on machines)) /m
33 for i=1 to m
34   Capacity(i)= ComputationalVolume   // setting the computation share of each machine
35 end for i
36 Assign=[ ]
37 for i=1 to length(sensitiveTasks)   // first phase
38   for j=1 to length(BestMachines for task i)
39     if (Capacity(BestMachines(j)>=ETC(i, BestMachines(j))
40       Assign(i)= BestMachines(j)
41       Capacity(BestMachines(j))= Capacity(BestMachines(j))- ETC(i,BestMachines(j))
42     else if among the tasks of BestMachines(j), one can be moved to free space for task i
43       move that task to another BestMachine
44       Assign(i)= BestMachines(j)
45       Capacity(BestMachines(j))= Capacity(BestMachines(j))- ETC(i,BestMachines(j))
46   end for j
47   if the task i has not yet been mapped to a machine
48     assign the machine with minimum runtime for task i and update the Capacity
49   end if
50 end for i
```

```
51  for each task i not assigned to a machine yet   // second phase
52    candidate_1= Machine with max(capacity)
53    candidate_2= Machine with min(ETC(task i))
54    candidate_3= Machine with the most parent or child tasks for the task i based on DAG D
55    if task i is a bottleneck one
56      candidate_4= Machine with fastest network connections
57      if candidate_1 is a SlowConnectionMachine
58        candidate_1=[ ]
59      end if
60      if candidate_2 is a SlowConnectionMachine
61        candidate_2=[ ]
62      end if
63    else
64      candidate_4=[ ]
65    end if
66    RT1 = Makespan of DAG D if  If the task i is mapped to candidate_1
67    RT2 = Makespan of DAG D if  If the task i is mapped to candidate_2
68    RT3 = Makespan of DAG D if  If the task i is mapped to candidate_3
69    RT4 = Makespan of DAG D if  If the task i is mapped to candidate_4
70    Assign task i to the machine with Min(RT1, RT2, RT3, RT4) and update its Capacity
71  end for
```

As can be seen, the proposed method considers the characteristics of resources and tasks, identifies sensitive and bottleneck tasks of the DAG and pays attention to the communicating capability of the machines. The resource allocation procedure is explained in more detail below.

On Lines (1-11), the algorithm finds sensitive tasks and their competent machines. As discussed in the previous section, the rows of the ETC matrix are sorted out in an ascending order first. Every row is then checked to see whether the runtime (t) of a task on a machine is twice or greater than that on the previous machine (Line 6). If true, that task is regarded as a sensitive task (T_S) and the previous machines (from the point on which the runtime is doubled) are considered competent for that task.

Lines (12-18) show how the average degrees of graph nodes are determined and how the bottleneck tasks are found. On Line 12, the average degree of the graph nodes is calculated. Line 15 determines whether there is a node the degree of which is three times or more than the average degree of the graph nodes. If there is a node matching the description, it is regarded as a bottleneck task (T_{bn}).

On Lines (19-29), the algorithm finds and determines fast and slow machines. In this section, the machine communication time matrix (CT_{ij}) is employed to determine the average time of communications between machines (Line 19). It is then checked whether the runtime of a machine is shorter than a half of the total average and if this applies, the machine is considered to be fast-connected. On the other hand, if the runtime is longer than twice the total average time, the machine is slowly-connected.

The next steps deal with how each task is allocated to the best machine such that the total runtime of the graph is reduced. The algorithm instructions are discussed in the following sub-sections. The main algorithm consists of two phases, the first of which is the allocation of sensitive tasks and the second phase is about the allocation of insensitive and bottleneck tasks.

4.2.1 First Phase: Allocation of Sensitive Tasks

First, the sensitive tasks are sorted in a descending order based on the average runtime of each task on Line 30. If also included among the bottleneck tasks, that sensitive task is not considered in this step (Line 31). This is because proper scheduling of bottleneck tasks is a key factor in reducing runtime in data-intensive workflows. As discussed earlier, since the balance between machines is a condition for the execution speed of tasks, Line 32 determines the computational load share of each machine to comply with the balance. In the next step, the sensitive tasks should be allocated to the best machine, which is performed in Lines 37-50. All of the sensitive tasks are sorted out in a descending queue and then the largest one is selected and allocated to the most

competent machine. Since the computational capacity (the computation share) of each machine was determined earlier, the runtime of a task should be subtracted from the computational load of that machine after the task is allocated, so that the available time of each machine is obtained to observe the balance condition (Line 41). If a common machine is calculated to be competent for several sensitive tasks, then the best case should be selected for allocation. For instance, if multiple sensitive tasks are already running on a machine, when a new task is to be assigned to the machine to complete the computational capacity of that machine, some of the previous tasks should be allocated to their other competent machines in the following way:

Regarding every sensitive task on that machine, it should be checked whether there is another competent machine. If so, the reduction in the runtime should be determined. After determining the reductions in runtimes for all sensitive tasks on that machine, the machine offering the smallest runtime is selected and the sensitive task experiencing the smallest reduction is allocated (moved) to its other competent machine. After performing this process, all of the sensitive tasks are scheduled on their competent machines. When all the sensitive tasks are finished, the remaining tasks are allocated in the next phase.

4.2.2 Second Phase: Allocation of Insensitive Tasks and Bottlenecks

Regarding every insensitive task, three machines are selected from the existing available machines: 1- the emptiest machine (the machine with the lowest allocated computational load), 2- the fastest machine (the machine which performs the task in the shortest period) and 3- the machine that hosts the largest number of parents or children of the intended task (Lines 52-54).

If the task is a bottleneck, there is also a fourth option: 4- the machine with fastest communications (Line 56). If a task is a bottleneck and the main share (more than a half) in its degree is due to its children, this task should be assigned to a machine with high communication capacity. Therefore, if the fastest and the emptiest candidate machines are slow in terms of communication, these two machines will be excluded from the list of candidates (57-62). Lines 66-70 determine on what candidates the insensitive and bottleneck nodes have the minimum runtime. The best machine is found as follows:

The graph is analyzed from the first node. Every insensitive and bottleneck task would be assigned to a candidate; i.e., the emptiest machine, the fastest machine, the machine with the largest number of parents and children or the machine with fast communications to determine the total runtime of the graph. The task is then assigned to the candidate offering the minimum makespan. The following steps are taken to determine the total runtime of a graph for each task on the candidate machines:

1. If tasks are already assigned to a machine on both sides of an edge, both the runtime and connection delay will be known.
2. If only one of the two-sided tasks of an edge is mapped, the average runtime of all machines for the unmapped task will be considered as an estimate for its runtime. Also, the average communication time between the mapped task's machine and other machines will be considered as an estimate for communication delay for this edge.
3. If no tasks are assigned on both sides of an edge, the runtime is considered the average runtime of all machines and the connection time is the average connection time of the entire graph.

The same procedure is followed to assign all tasks to their corresponding proper machines offering the minimum workflow makespan. In this study, fast-connected machines were employed to execute the bottleneck tasks given their large number of connections required; i.e., parents and children, so that the runtime could be optimized by reducing the communication delays. The makespan of the workflow is determined by finding the critical path of the graph.

4.2.3 Time Complexity of the Algorithm

Based on the pseudocode provided, it can be stated that most of the computations occur in nested loops of lines 37 to 50. In the worst case, if all the tasks are sensitive and all the machines are

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 07, No. 04, December 2021.

available for mapping, according to the search performed in the body of these loops, we can say that the algorithm of the proposed method has a complexity of the order of $O(m \times n \times \log n)$. Parameter m indicates the number of available machines and parameter n indicates the number of jobs.

5. SIMULATION RESULTS

This section describes the performance evaluation of the proposed method. First, the benchmark workflows used in the experiments are described and then the system simulation conditions and configuration are detailed. Finally, the results of evaluation and performance analysis of the proposed method in comparison with GSS [23], CAS-L1 [31] and MCC [32] are described.

5.1 Benchmark Workflows

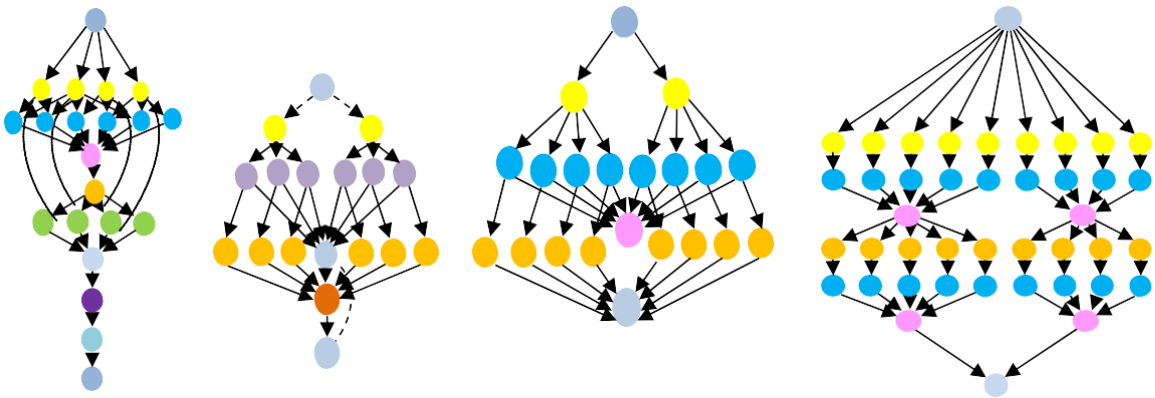
The workflows used in the experiments are generated based on real applications' structures. There are four types of workflow applications named Cybershake, Epigenomics, Montage and Inspiral (LIGO), which are used extensively for generating synthetic workflows in previous related research. Figure 2 shows these workflows:

1. Cybershake: This application is utilized to describe the hazards of earthquakes in a specific region by using the seismic hazard curve.

2. Epigenomics: This application is employed to show the epigenetic status of human cells on a large scale in genome.

3. Montage: This application was developed by NASA/IPAC to generate the aerial input images in a customized tiled format.

4. Inspiral (LIGO): This application was based on Einstein's theory for the detection of gravity waves.



(a) (b) (c) (d)

Figure 2. The structures of benchmark workflow applications [24].

More details of these applications can be found in [34][35]. Based on these structures, 400 synthetic workflows are generated using [35], 100 instances of each type. Since these structures are derived from real applications, it helps to have a fairer evaluation than using graphs that are made completely at random. Since this research focuses on data-intensive workflows and to better evaluate the proposed method, these workflows have been created from various CCR (communication to computation ratio) values (from 0.3 to 0.7). Higher values of the communication to computation ratio correspond to more intensives workflows. Task compute amounts and data transfer sizes are generated randomly with uniform distribution.

5.2 Simulated System Configuration

Cloudsim simulator [36] is used to model a realistic computing system and network. The system includes a cluster of 10 processing elements each one having 8 computing cores.

Therefore, 8 virtual machines can be run in parallel on each physical host. The computation power of these machines is set based on the characteristics of real-world cloud systems (i.e., Amazon's EC2). These machines are connected with Gigabit Ethernet network. However, communication links are considered to have different delays, which are determined randomly in the range of 10 to 100 milliseconds.

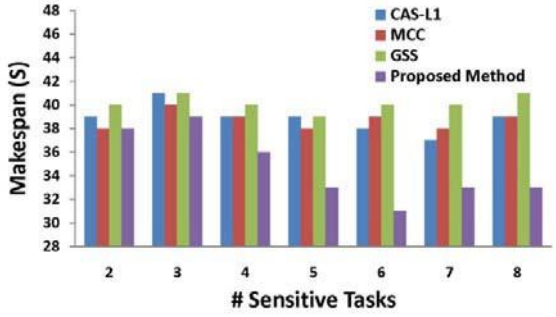
In the following part, the experiments performed are described and their results are analyzed. In order to examine the effect of defined concepts, such as sensitive work and bottlenecks, an attempt has been made to examine the performance of the proposed method in different cases of the number of sensitive tasks and the heterogeneity of resources in the system. The experiments described in the next sub-section were repeated 10 times and the mean values obtained were reported as a more reliable result.

5.3 Experimental Results

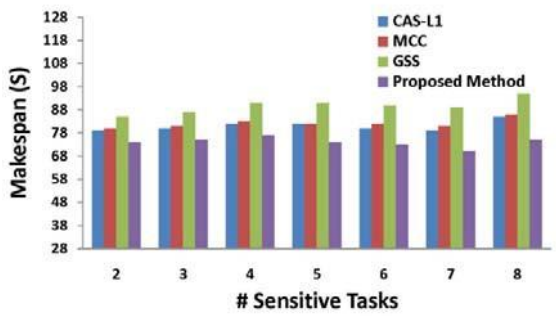
As discussed earlier, paying attention to the structure of workflows and considering the fact that one machine may be good for running one program while being not well for another, the proposed scheduling method is distinguished from previous ones. In fact, looking at the features of the task and the machine separately cannot always lead to proper mapping and one must look at the program and machine together. The concept of sensitive task defined in this research wants to say that among the available resources, some machines may be much more suitable for performing some tasks than others and this issue should be a priority in mapping tasks to resources. Three factors are considered in the performance of the proposed method and its performance is compared with previous work in terms of: the number of sensitive tasks, the number of employed machines and the heterogeneity or homogeneity of resources.

5.3.1 Different Numbers of Sensitive Tasks

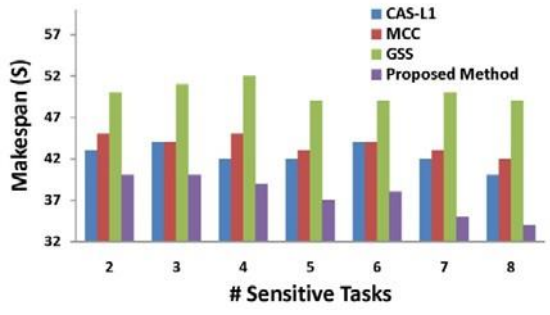
In the first scenario, the number of sensitive tasks of each graph was changed in the workflows when the number of machines was constant. These changes can be controlled when setting the runtimes of each workflow on each virtual machine (the ETC matrix). Figure 3 shows the experimental results for performance evaluation of the proposed method.



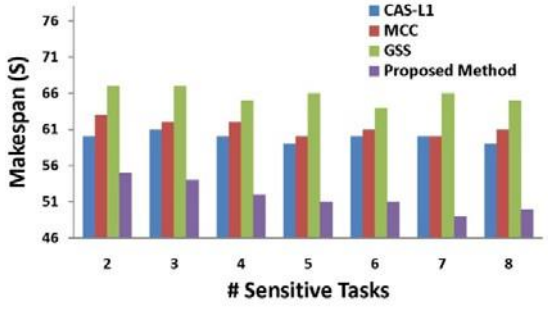
Graph (a) with 5 virtual machines



Graph (b) with 7 virtual machines



Graph (c) with 8 virtual machines



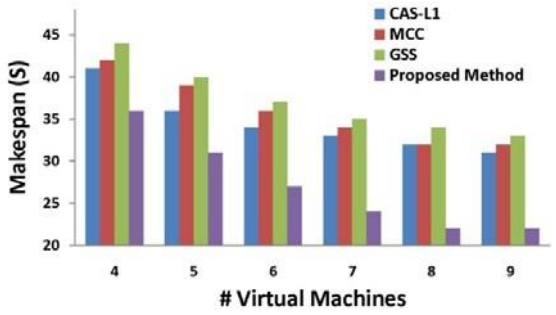
Graph (d) with 12 virtual machines

Figure 3. The performance of the proposed method in comparison with previous related works for different numbers of sensitive tasks in the workflows.

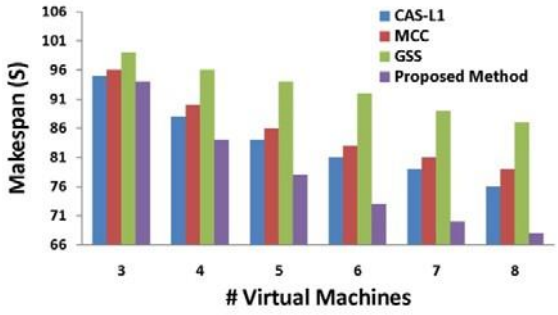
According to the results, a reduction in the execution time of workflows for the proposed method is visible compared to other methods. Paying attention to sensitive and vulnerable tasks and the priority in scheduling these tasks has led to this improvement. As the number of sensitive tasks in the workflow increases, the superiority of the proposed method in improving execution time becomes more apparent.

5.3.2 Different Numbers of Virtual Machines

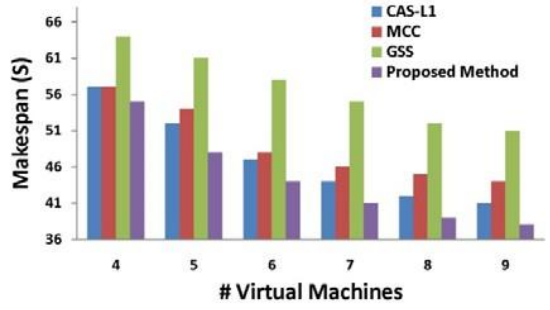
In the second scenario, the performance of the proposed method in comparison with other methods is evaluated for different numbers of virtual machines. Other factors such as the number of sensitive tasks and the data-intensive nature of workflows are considered the same. Figure 4 shows the results for each type of workflow structure.



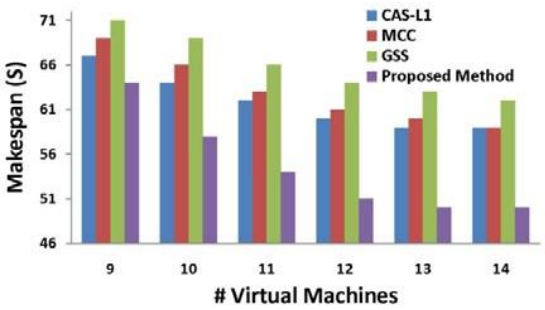
Graph (a) with 6 sensitive tasks



Graph (b) with 7 sensitive tasks



Graph (c) with 4 sensitive tasks



Graph (d) with 6 sensitive tasks

Figure 4. The performance of the proposed method in comparison with previous related works for different numbers of virtual machines.

Based on the results, it can be said that regardless of the number of virtual machines used, the proposed method shows better performance compared to similar previous methods. This may be due to the fact that the proposed method pays special attention to the structure of workflows, bottlenecks and sensitive tasks. The increase in the number of virtual machines in general has further improved the performance of the proposed method and this may be due to more options for scheduling decisions.

5.3.3 The Degree of Resource Heterogeneity

In the third scenario, an attempt has been made to evaluate the performance of the proposed method at different levels of resource heterogeneity. The results are presented in three categories: homogeneous, heterogeneous and moderate heterogeneity. It is expected that as the heterogeneity in resources increases, the performance of the proposed method will also improve. Since the proposed method pays attention to the fact that some resources have special capabilities to run some tasks, so increasing heterogeneity while increasing the options for scheduling, the chances of such capabilities among the available machines for execution of a task also increase.

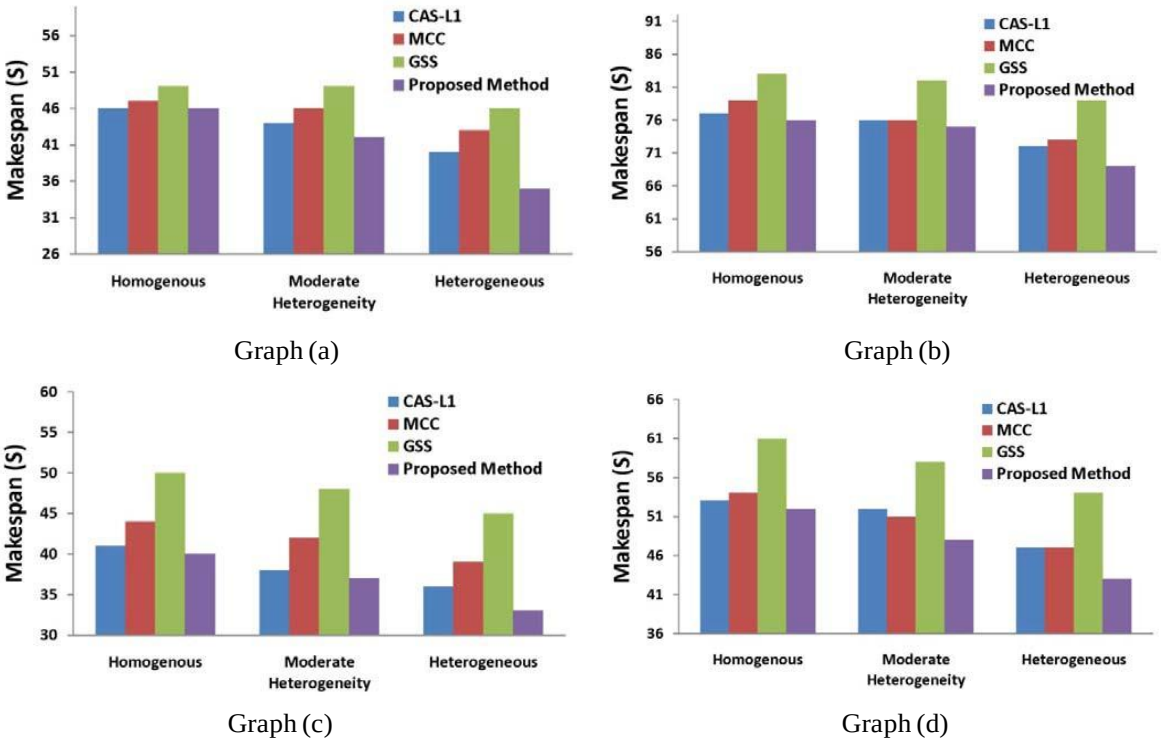


Figure 5. The performance of the proposed method in comparison with previous related works at different levels of resource heterogeneity.

As can be seen from the results, the proposed method has been successful in reducing the execution time of data-intensive workflows compared to similar previous works. Attention to the graph structure of workflows, detection of bottlenecks and sensitive programs and allocation of appropriate resources to them with higher priority, attention to different features of machines in communication delays and balancing computational load can be mentioned as reasons for this performance.

6. Conclusions and Future Work

Applications requiring big data have now gained more importance as their usage continues to grow. Therefore, it is necessary to focus on their scheduling methods and consider their differences to manage them better. In the proposed method, a heuristic scheduling algorithm with a low time complexity was introduced to minimize the workflow makespan by analyzing workflows' graph structure and the tasks' runtime on machines of different physical capabilities. This procedure also considered that all the data required by a task might be read from the database in a single attempt. Additionally, the connections between machines were taken into account to find fast-connected machines for those tasks requiring more data transformation; i.e., bottleneck tasks, in order to reduce the makespan of data-intensive workflows. The proposed method, based on the results, has had a more successful performance in reducing the total time required to execute workflows compared to other previous methods.

In future work, it is possible to deal with the interference of virtual machines located on one host on each other's performance and by modeling these effects to incorporate them into decisions. Also, considering the chance of machine failure when scheduling to increase system reliability can be another effective factor in scheduling decisions. Reducing energy consumption along with workflow execution time can turn the scheduling problem into a two-criterion optimization problem.

References

[1] Y. Ahn and Y. Kim, "Auto-scaling of Virtual Resources for Scientific Workflows on Hybrid Clouds," Proc. of the 5th ACM Workshop on Scientific Cloud Computing (ScienceCloud '14), pp. 47-52, DOI: 10.1145/2608029.2608036, June 2014.

[2] L. F. Bittencourt and E. R. M. Madeira, "HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds," Journal of Internet Services and Applications, vol. 2, pp. 207-227, 2011.

[3] S. Sagiroglu and D. Sinanc, "Big Data: A Review," Proc. of the IEEE International Conference on Collaboration Technologies and Systems (CTS), pp. 42-47, San Diego, CA, USA, July 2013.

[4] K. Wang, K. Qiao, I. Sadooghi, X. Zhou, T. Li, M. Lang et al., "Load-balanced and Locality-aware Scheduling for Data-intensive Workloads at Extreme Scales," Concurrency and Computation: Practice and Experience, vol. 28, pp. 70-94, 2016.

[5] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," Proc. of the 5th European Conf. on Computer Systems (EuroSys '10), pp. 265-278, DOI: 10.1145/1755913.1755940, April 2010.

[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," Proc. of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI'08), vol. 8, no. 4, pp. 29-42, December 2008.

[7] B. Lin, W. Guo and X. Lin, "Online Optimization Scheduling for Scientific Workflows with Deadline Constraint on Hybrid Clouds," Concurrency and Computation: Practice and Experience, vol. 28, pp. 3079-3095, August 2016.

[8] N. Xiong, X. Jia, L. T. Yang, A. V. Vasilakos, Y. Li and Y. Pan, "A Distributed Efficient Flow Control Scheme for Multirate Multicast Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 9, pp. 1254-1266, September 2010.

[9] J. Yin, X. Lu, X. Zhao, H. Chen and X. Liu, "BURSE: A Bursty and Self-similar Workload Generator for Cloud Computing," IEEE Trans. on Parallel and Distributed Sys., vol. 26, no. 3, pp. 668-680, 2015.

[10] Y. E. M. Hamouda, "Modified Random Bit Climbing (λ -mRBC) for Task Mapping and Scheduling in Wireless Sensor Networks," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 5, no. 1, pp. 17-32, April 2019.

[11] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos and H. Zhang, "A Pretreatment Workflow Scheduling Approach for Big Data Applications in Multicloud Environments," IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 581-594, September 2016.

[12] A. N. Toosi, R. O. Sinnott and R. Buyya, "Resource Provisioning for Data-intensive Applications with Deadline Constraints on Hybrid Clouds Using Aneka," Future Generation Computer Systems, vol. 79, no.2, pp. 765-775, February 2018.

[13] M. Sohani and S. C. Jain, "Fault Tolerance Using Self-healing SLA and Load Balanced Dynamic Resource Provisioning in Cloud Computing," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 07, no. 02, pp. 206-222, June 2021

[14] G. L. Stavrinides, F. R. Duro, H. D. Karatza, J. G. Blas and J. Carretero, "Different Aspects of Workflow Scheduling in Large-scale Distributed Systems," Simulation Modeling Practice and Theory, vol. 70, pp. 120-134, January 2017.

[15] M. Adhikari and T. Amgoth, "Multi-objective Accelerated Particle Swarm Optimization Technique for Scientific Workflows in IaaS Cloud," Proc. of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1448-1454, Bangalore, India, September 2018.

[16] H. G. E. D. H. Ali, I. A. Saroit and A. M. Kotb, "Grouped Tasks Scheduling Algorithm Based on QoS in Cloud Computing Network," Egyptian Informatics Journal, vol. 18, no. 1, pp. 11-19, March 2017.

[17] S. Suresh, H. Huang and H. J. Kim, "Scheduling in Compute Cloud with Multiple Data Banks Using Divisible Load Paradigm," IEEE Transactions on Aerospace and Electronic Systems, vol. 51, no. 2, pp. 1288-1297, 2015.

[18] M. Kowsigan and P. Balasubramanie, "Scheduling of Jobs in Cloud Environment Using Soft Computing Techniques," Int. Journal of Applied Engineering Research, vol. 10, no. 38, pp. 28640-28645, 2015.

[19] W. Yan, W. Jinkuan and H. Yinghua, "Cloud Computing Workflow Framework with Resource Scheduling Mechanism," Proc. of the IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), pp. 342-345, Nanjing, China, August 2016.

[20] P. Kaur and S. Mehta, "Resource Provisioning and Workflow Scheduling in Clouds Using Augmented Shuffled Frog Leaping Algorithm," Journal of Parallel and Distributed Computing, vol. 101, pp. 41-50, 2017.

[21] J. Shamsi, M. A. Khojaye and M. A. Qasmi, "Data-intensive Cloud Computing: Requirements, Expectations, Challenges and Solutions," Journal of Grid Computing, vol. 11, pp. 281-310, April 2013.

[22] S. G. Ahmad, C. S. Liew, M. M. Rafique and E. U. Munir, "Optimization of Data-intensive Workflows in Stream-based Data Processing Models," The Jour. of Supercomputing, vol. 73, pp. 3901-3923, 2017.

[23] M. S. Kumar, I. Gupta, S. K. Panda and P. K. Jana, "Granularity-based Workflow Scheduling Algorithm for Cloud Computing," The Journal of Supercomputing, vol. 73, pp. 5440-5464, June 2017.

[24] F. Xiong, C. Yeliang, Z. Lipeng, H. Bin, D. Song and W. Dong, "Deadline Based Scheduling for Data-Intensive Applications in Clouds," The Journal of China Universities of Posts and Telecommunications, vol. 23, no. 6, pp. 8-15, December 2016.

[25] T. Ghafarian and B. Javadi, "Cloud-aware Data Intensive Workflow Scheduling on Volunteer Computing Systems," Future Generation Computer Systems, vol. 51, no. C, pp. 87-97, October 2015.

[26] K. Kanagaraj and S. Swamynathan, "Structure Aware Resource Estimation for Effective Scheduling and Execution of Data Intensive Workflows in Cloud," Future Generation Computer Systems, vol. 79, no. P3, pp. 878-891, February 2018.

[27] S. Esteves and L. Veiga, "WaaS: Workflow-as-a-Service for the Cloud with Scheduling of Continuous and Data-intensive Workflows," The Computer Journal, vol. 59, no. 3, pp. 371-383, March 2016.

[28] I. Pietri and R. Sakellariou, "Scheduling Data-intensive Scientific Workflows with Reduced Communication," Proc. of the 30th International Conference on Scientific and Statistical Database Management (SSDBM '18), pp. 1-4, DOI: 10.1145/3221269.3221298, July 2018.

[29] P. Wang, Y. Lei, P. R. Agbedanu and Z. Zhang, "Makespan-driven Workflow Scheduling in Clouds Using Immune-based PSO Algorithm," IEEE Access, vol. 8, pp. 29281-29290, February 2020.

[30] G. Ismayilov and H.R. Topcuoglu, "Neural Network Based Multi-objective Evolutionary Algorithm for Dynamic Workflow Scheduling in Cloud Computing," Future Generation Computer Systems, vol. 102, pp. 307-322, January 2020.

[31] O. Sukhoroslov, "Toward Efficient Execution of Data-intensive Workflows," The Journal of Supercomputing, vol. 12, pp. 7989-8012, 2021.

[32] F. Li, "A Novel Scheduling Algorithm for Data-intensive Workflow in Virtualized Clouds," International Journal of Networking and Virtual Organizations, vol. 20, no. 3, pp. 284-300, June 2019.

[33] H. Saadatfar and H. Deldari, "A Study on Combinational Effects of Job and Resource Characteristics on Energy Consumption," Multiagent and Grid Systems, vol. 9, no. 4, pp. 301-314, January 2014.

[34] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta and K. Vahi, "Characterizing and Profiling Scientific Workflows," Future Generation Computer Systems, vol. 29, no. 3, pp. 682-692, March 2013.

[35] Confluence, "Workflow Generator," [Online], Available: https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

[36] T. Goyal, A. Singh and A. Agrawal, "Cloudsim: Simulator for Cloud Computing Infrastructure and Modelling," Procedia Engineering, vol. 38, pp. 3566-3572, DOI: 10.1016/j.proeng.2012.06.412, 2012.

ملخص البحث:

تقترح هذه الورقة خوارزمية جدولةٍ جديدةً لتدفّقات العمل كثيفة البيانات بناءً على اعتمادية البيانات في السّحابات الحاسوبية. وتحاول الخوارزمية المقترحة التقليل الى الحدّ الأدنى من نطاق العمل مع اعتبار تفاصيل بُنْية تدفّق العمل والآلات الافتراضية. وتجدر الإشارة الى أنّ المفاهيم والتفاصيل التي تمّ تعريفها وأخذها بعين الاعتبار في الخوارزمية المقترحة لم تنلْ ما يكفي من الاهتمام في أبحاثٍ سابقةٍ مماثلة.

وبناءً على النتائج، فقد قلّلت الخوارزمية المقترحة زمن الاتّصال بين المهمات وفترات التشغيل عن طريق أخذ سِمات تدفّقات العمل كثيفة البيانات والتّخصيص المحْكم للمهمات بعين الاعتبار. وبالنتيجة، خفّضت الخوارزمية المقترحة النّطاق الإجمالي للعمل مقارنةً بالخوارزميات الواردة في أعمالٍ سابقةٍ حول الموضوع.