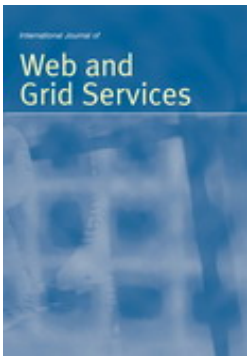
INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research



International Journal of Web and Grid Services

ISSN online: 1741-1114 - ISSN print: 1741-1106
https://www.inderscience.com/ijwgs

Data locality-aware and QoS-aware dynamic cloud workflow scheduling in Hadoop for heterogeneous environment

Fan Ding, Minjin Ma

DOI: 10.1504/IJWGS.2023.10054497

Article History:

Received:	10 January 2022
Last revised:	26 November 2022
Accepted:	04 December 2022
Published online:	06 March 2023

Copyright © 2023 Inderscience Enterprises Ltd.

Data locality-aware and QoS-aware dynamic cloud workflow scheduling in Hadoop for heterogeneous environment

Fan Ding*

College of Computer and Communication Engineering,
Lanzhou University of Technology,
No. 287 Langongping Road, Qilihe District,
Lanzhou City, Gansu, 730050, China
Email: dingf14@163.com
*Corresponding author

Minjin Ma

College of Atmospheric Sciences,
Lanzhou University,
No. 222 South Tianshui Road, Lanzhou 730000,
Gansu Province, 730000, China
Email: minjinma@lzu.edu.cn

Abstract: Hadoop has become a popular data-parallel computing framework for data-intensive scientific applications in recent years. Most scientific applications employ workflows to portray procedures and dependencies between jobs. However, the current default scheduling policy in Hadoop does not take data locality into account. The movement of data among virtual machines (VMs) produces latency in workflow scheduling. In addition, the heterogeneous and dynamics of cloud resources cannot satisfy the user's demand for quality of service (QoS) in static workflow scheduling. Hence, we propose a data locality-aware and QoS-aware dynamic cloud workflow scheduling algorithm (DQ-DCWS) based on dynamic programming. The algorithm balances data locality and delays by grouping nodes that hold tasks correlated with data blocks. We consider five QoS factors and normalise them as a path optimisation issue to realise maximum QoS. DQ-DCWS is implemented and validated by running Montage workflow on real Hadoop clusters which are deployed on Amazon EC2.

Keywords: data locality; Hadoop MapReduce; heterogeneous; workflow scheduling; quality of service; QoS; big data.

Reference to this paper should be made as follows: Ding, F. and Ma, M. (2023) 'Data locality-aware and QoS-aware dynamic cloud workflow scheduling in Hadoop for heterogeneous environment', *Int. J. Web and Grid Services*, Vol. 19, No. 1, pp.113–135.

Biographical notes: Fan Ding received her BS and PhD in Computing Science from Lanzhou University, in 2008 and 2014, respectively. Currently, she is an Associate Professor in the College of computer and Communication Engineering at Lanzhou University of Technology. Her current research interest includes cloud computing, big data parallel and distributed processing and machine learning.

Copyright © 2023 Inderscience Enterprises Ltd.

Minjin Ma is an Associate Professor in the Lanzhou University. He graduated from the Lanzhou University as a Bachelor degree in 2005. He obtained his doctorate degree (PhD) in the College of Atmospheric Sciences, Lanzhou University in 2011. His major research field is atmospheric boundary layer, urban air pollution and numerical simulation. His current interesting is to improve weather and air quality forecast by new technology such as image recognition, machine learning and big data. He is kind at computer programming and numerical simulation. Additionally, he has special experience of TRIZ training and now become an Innovation Engineer and a researcher in Gansu Innovation Methodology Research Centre.

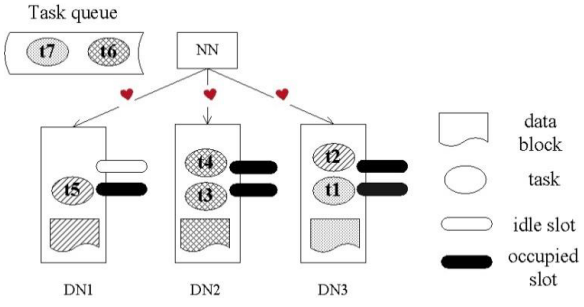
1 Introduction

In the era of big data, data-intensive applications have attracted substantial attention from researchers in a diverse range of fields, such as environmental science (Davis et al., 2017), astronomy, bioinformatics (Mondelli et al., 2018) and materials science (Service, 2012). Scalability and large-scale resources provided by the cloud enable data-intensive application efficiency. Scientists usually employ science workflow (SWf) in data-intensive applications. SWf is a flexible tool that acquires and conducts a complex analysis of scientific data, including sensor data, medical images, simulator outputs, satellite images, and various observation data. SWf automatically composes and performs a complex analysis of distributed resources for data-intensive applications in cloud computing. There are many popular SWf systems, such as Pegasus (Deelman et al., 2021), KNIME (Lumley et al., 2020), Galaxy (Afgan et al., 2018), Kepler (Owsiak et al., 2017), Taverna (Kasztelnik et al., 2017), and Triana (Zhang et al., 2011). These workflow systems aim to automate scientists' complex work in the computing environment and release them from the bondage of tools and computation skills.

In the cloud computing environment, data-intensive applications must cope with many challenging and crucial issues. Users submit a data analytics job on a workflow system, and then, the job is decomposed into many tasks that are scheduled on separate virtual machines (VMs). Importantly, large scale data processing demands a parallel programming framework. Hadoop (White, 2010) is an open-source framework developed by Apache for the storage and processing of large datasets that employ MapReduce (Dean and Ghemawat, 2004) to parallel computing jobs with the datasets. Hadoop has been deployed in many leading companies (e.g., Facebook, Yahoo and Baidu), mainly for big data applications such as data indexing, web analytics, document processing, etc. It can be built quickly on hundreds of nodes and enables highly scalable big data processing architectures.

Hadoop Mapreduce Version 2 (Hadoop Yarn) is currently Hadoop framework including resource manager, application master and node manager. Yarn schedule Mapreduce job on Hadoop distributed file system (HDFS) which consists of one NameNode (NN) as master and many DataNodes (DN) as workers. MapReduce executes a set of map and reduce tasks concurrently to improve the execution efficiency on DN. The resource manager runs on NN that manages all nodes, including master nodes and worker nodes, and maintains a scheduler to allocate tasks to idle nodes.

Figure 1 Hadoop MapReduce scheduling and data locality (see online version for colours)



The scheduling procedure is shown in Figure 1. While starting a job on Hadoop, the application master schedules the tasks to worker nodes from a task queue, and the node manager monitors the running state of the tasks by heartbeat information (represented as red hearts in Figure 1). The NN reports the slot information, which refers to whether a node has a place to run a new task on the VM. The Hadoop scheduler is based on the slot, which is configured to run map/reduce tasks concurrently. The number of slots on a node is limited by default, as two consist of one map slot and one reduced slot. Slots are occupied by tasks one by one until no slots are available. Meanwhile, the DN periodically sends a heartbeat, which tells the status of the current DN to the NN. Once the node has an idle slot, the idle slot pulls a task from the job queue. Hadoop divides input files into one or more data blocks that are configured as the default 128 MB for each block. Data locality is an important factor in the scheduling process. Data locality indicates that a task is scheduled on a node that places its correlated data block. Optimal scheduling ensures data locality. The idle slot gives priority to the task which is correlated to the data blocks placed on the node. If the slot cannot find that kind of task, then the task-correlated data blocks will be moved to the node to be executed so that the result will be delayed, which means that data locality at the node level is not implemented. As shown in Figure 1, the two tasks on DN2 realise data locality. However, DN3 needs to pull the data block correlated with task t2 from DN1. Thus, the lack of data locality leads to additional data migration time and communication cost on cloud resources. Therefore, current MapReduce scheduling is not optimal in heterogeneous cloud environments. Two factors contribute to the lack of data locality in the current Hadoop implementation:

1 The default data placement policy in Hadoop is simple and random (Borthakur, 2008) and does not place correlated data blocks on different nodes. If all correlated data are placed on the same node, then some tasks must wait for the other running tasks to be completed when there are multiple tasks to be performed.

2 The scheduling policy in MapReduce includes FifoScheduler, FairScheduler and CapacitySchedule (Dean and Ghemawat, 2004). Although these algorithms can realise fair and capacity scheduling, they cannot ensure data locality. As shown in Figure 1, when DN1 has an idle slot and sends heartbeat information to the NN, the NN first look for a task that has a correlated data block placed on DN1. However, the remaining two tasks in the task queue are not related to the data block in DN1. Thus, the NN randomly selects a task and allocates it to DN1. Data locality is not realised.

Cloud SWf scheduling must consider not only data locality but also other factors, including system workload, overall fairness and the communication cost produced by intermediate data between two related tasks. Moreover, the heterogeneity of VM resources leads to a different quality of service (QoS) for cloud nodes. Traditional SWf scheduling uses static scheduling approach, which can find the most optimal scheduling solution in the homogeneous resource environment. However, it becomes no longer applicable in heterogeneous cloud resource environments where multi-QoS requirements are considered. According to the above mentioned arguments, the SWf scheduling of Hadoop in a heterogeneous environment faces the following challenges.

1 How to achieve data locality in workflow scheduling and minimise the latency from data movement?

2 How to dynamically select VM resources to achieve the optimal workflow scheduling?

3 How to minimise the cost of cloud resources while maintaining the QoS in wrokflow scheduling?

In our work, we propose a novel approach, data locality-aware and QoS-aware dynamic cloud workflow scheduling (DQ-DCWS), which considers the data locality and QoS of VMs in heterogeneous environments based on a dynamic programming method. DQ-DCWS balances data locality and latency by grouping nodes that hold correlated data blocks with tasks. We model the issue as a path optimisation to realise the maximum QoS. We implement and validate a dynamic cloud scheduling algorithm by running a real workflow on deployed Hadoop clusters over Amazon Elastic Compute Cloud (EC2) (Amazon.com, 2022).

The rest of the paper is organised as follows. Section 2 provides the important related works. Section 3 details problem definition for SWf and data locality in Hadoop. Section 4 presents details for the proposed data locality-aware and QoS-aware dynamic cloud workflow scheduling. Section 5 presents detailed algorithm comparisons and experimental analysis. Followed by the conclusions and future directions in Section 6. Finally, the conclusions and future work are included in Section 6.

2 Related works

A wide variety of research has addressed optimisation of Hadoop scheduling through different aspects, including cost-effective workflow scheduling (Rashmi and Basu, 2016; Fang and Sun, 2018; Nasr et al., 2018; Thingom et al., 2018; Javanmardi et al., 2021), performance optimisation based on task partitioning (Tong and Wu, 2017; Zhang et al., 2017), storage-aware workflow scheduling (Ye et al., 2018), and multi-QoS constrained workflow scheduling (Zheng et al., 2017; Arabnejad and Barbosa, 2017; Seth and Singh, 2020; Qureshi, 2019). Other methods (Xu and Yong, 2015; Rashmi and Basu, 2017; Alwidian and Alaa Abdallat, 2019; Dey and Gunasekhar, 2019; Gandomi et al., 2019; Seethalakshmi et al., 2020; Kalia et al., 2022) apply existing optimisation algorithms to improve scheduling efficiency. One of the most widely used algorithms is the particle swarm optimisation (PSO) algorithm (Fang and Sun, 2018; Garg and Singh, 2014; Hu et al., 2007; Li et al., 2015; Verma and Kaushal, 2014; Wu et al., 2011; Kchaou et al., 2022), which is an evolutionary algorithm based on swarm intelligence theory. The PSO

(Kennedy, 1995) algorithm modifies the velocity and position of the particles by acquiring the relevant memory particles. In workflow scheduling, PSO enables fast convergence and has the advantage of being both simple and effortless.

This work focuses on the data locality and heterogeneity of the resources mentioned in introduction. Two main aspects are considered to improve the overall performance of SWf scheduling in cloud.

2.1 Data placement strategy

An appropriate data placement strategy can reduce the scheduling overhead. The objective of the data placement policy is to place correlated task data on different nodes in a heterogeneous environment. In a previous study (Wang et al., 2013); a DRAW scheme considering data grouping semantics was developed. Data grouping refers to data that can be accessed simultaneously. The frequency of access is proportional to the correlation of the data. A matrix is employed to model the correlation of the data. The top two groups of correlated data are computed by converting them into a clustering matrix. This method maximises the even distribution of correlated data and balances the storage loads. Another work (Wu et al., 2016) has made an improvement based on that proposed in Wang et al. (2013), which concerns the execution frequency of jobs. However, the authors ignore the fact that tasks need to wait for occupied slots and whether running tasks are related to the blocks of the same grouping data in the operation peak period. There are many idle slots waiting for another group of correlated data. Thus, the procedure produces disuse and wastes resources. Singh et al. (2018) proposed a data placement technique for fixed and non-fixed datasets in a cloud environment that improves the overall makespan time for workflow execution. Kim and Kim (2018) proposed an adaptive data placement strategy considering dynamic resource changes for efficient data-intensive applications. Another work (Qian et al., 2017) presented an adaptive data placement approach that dynamically adjusts to asynchronous coupling patterns. This approach places data across a set of staging nodes with an awareness of the application-specific data locality requirements and runtime task executions at these staging nodes.

Most Hadoop-based systems make block-data distribution and resource allocation independent. A scheduling scheme (DPPACS) (Reddy and Roy, 2018) exploits knowledge of data availability at different clusters. A dynamic data placement strategy (DDPS) has been proposed to obtain the best location for new replicas according to their hotness (Liu et al., 2018a). The method is able to allocate data replicas dynamically and reduce the response time in a heterogeneous environment. The strategy presented in Vengadeswaran and Balasundaram (2019) redistributes data blocks to produce an optimal data placement after dynamically analysing the history log and establishing the relationship between various tasks and blocks required for each task through a block dependency graph (BDG). An entropy-based grouping technique was investigated in Reddy et al. (2019). The authors grouped datasets with the most similar existing clusters based on their relative entropy. Another data placement strategy (Bae et al., 2020) in Hadoop preserves the same degree of data locality with a small number of replicas, which select and copy only the data blocks that is most likely to be accessed remotely.

There is also a situation in which an input split consists of many data blocks that are distributed in different nodes. Increasing the data block replication frequency will slow

down overall performance. Choi et al. (2018) classified data locality by considering the locations of all data blocks that contain an input split and categorising tasks.

2.2 Workflow scheduling policies

Most workflow scheduling works to optimise efficiency assuming that cloud services are homogeneous. However, in heterogeneous environments, the data blocks of the Hadoop clusters are allocated equally to all nodes without considering the different capabilities of individual nodes. This situation results in low performance in Hadoop scheduling. The existing static algorithms are unable to resolve issues in heterogeneous environments. Guo et al. (2012) proposed a static workflow scheduling algorithm, lsap-sched that allocates all tasks on idle slots. Lsap-sched weighs each task slot according to whether it has correlational data, and then, it computes the task slot of minimising the weight with a transferred matrix model. At the beginning of the scheduling period, it shows high performance due to the large number of unscheduled tasks and idle slots, so data locality is obtained. However, the superiority of the algorithm is not obvious when heavy tasks need to be scheduled, because few slots are idled. Therefore, it is important to balance data locality and scheduling latency. Mon et al. (2017) presented a clustering method-based task dependency to reduce execution overhead and improve the computational granularity of SWf tasks. In Wylie et al. (2016), both an optimal and a heuristic approach to satisfy the minimum makespan and a given budget constraint on the Hadoop MapReduce platform were employed. Zhou et al. (2019) selected the node as the optimal node with the highest matching degree, which considers the distance among nodes, the load of nodes and the number of backup data recoveries. This strategy realises load balancing and considers the internal bandwidth consumed during data recovery. Liu et al. (2018b) proposed an adaptive discrete PSO algorithm based on a genetic algorithm to decrease the number of data transmissions across data centres. Furthermore, in Manasrah and Ali (2018), a hybrid GA-PSO algorithm to allocate tasks to resources efficiently was proposed. The algorithm aims to reduce the makespan and cost.

Some of the most recent related studies are based on dynamic SWf scheduling methods. In one article (Soualhia et al., 2020), the researchers presented a dynamic and failure-aware framework that can be integrated within the Hadoop scheduler and adjust the scheduling decisions based on collected information about the cloud environment. Hu et al. (2018) provided a framework designed to make scheduling decisions for workflows to meet deadlines and simultaneously optimises the performance of ad hoc jobs. Wang et al. (2017) proposed a workflow scheduling algorithm with a dynamic priority focus on deadlines to achieve more reasonable fairness.

As shown in Table 1, the related work have be divided into two categories, one only taking into account data placement (Wang et al., 2013; Wu et al., 2016; Singh et al., 2018; Kim and Kim, 2018; Qian et al., 2017; Reddy and Roy, 2018; Liu et al., 2018a; Vengadeswaran and Balasundaram, 2019; Reddy et al., 2019; Bae et al., 2020) and the other including workflow scheduling like (Choi et al., 2018; Guo et al., 2012; Mon et al., 2017; Wylie et al., 2016; Manasrah and Ali, 2018; Soualhia et al., 2020; Hu et al., 2018; Wang et al., 2017). A few literatures consider the two aspects simultaneously (Kchaou et al., 2022; Liu et al., 2018b; Zhou et al., 2019).

Different from the existing work, we investigate the workflow scheduling procedure rests on data placement based on data-locality in addition to QoS in heterogeneous environments of Hadoop-based systems. Besides, in the proposed strategy, more attention

is drawn to dynamic scheduling based on dynamic programming algorithm instead of approaches based on static scheduling, which dynamic plan the optimal path in workflow scheduling to minimise the SWf cost and execution time.

Table 1 Research on data placement strategy and scheduling policies of Hadoop workflow scheduling

Approach	*Metrics (QoS)*	*Data placement*	*Dynamic*	*Optimisation objectives*
FPSO (Kchaou et al., 2022)	Volume of data transfer	√	×	Reducing the data transmission cost
DRAW (Wang et al., 2013)	Storage loads	√	×	Achieve the maximum parallelism per data-group
DGAD (Wu et al., 2016)	Makespan time			
Heuristic data placement (Singh et al., 2018)	Overhead	√	×	Access data at the lowest cost of data transfer
Adaptive data placement (Kim and Kim, 2018; Qian et al., 2017)	Makespan	√	×	Reduce data movement in workflow
DPPACS (Reddy and Roy, 2018)	Execution time	√	×	Improving computation and data availability
DDPS (Liu et al., 2018a)	Execution time	√	√	Dynamically adjust replicas location according to hotness
CORE (Vengadeswaran and Balasundaram, 2019)	Execution time	√	×	Optimise data placement based on relationship between tasks and blocks
Entropy-based data placement (Reddy et al., 2019)	Overhead	√	√	Grouped datasets with the most similar existing clusters.
Data-placement scheme (Bae et al., 2020)	Overhead	√	×	Preserves same degree of data locality with a small number of replicas.
Data-locality scheduling (Choi et al., 2018)	Makespan	-	×	Classified data locality
Lsap-sched (Guo et al., 2012)	Execution time	-	×	Schedules multiple tasks simultaneously to give optimal data locality
Clustering method (Mon et al., 2017)	Overhead	-	×	Clustering method-based task dependency
Budget-constrained scheduling (Wylie et al., 2016)	Makespan Budget constraint	-	×	Minimise workflow makespan while satisfying a given budget constraint.
Backup data placement policy (Zhou et al., 2019)	Load balancing	√	×	Reduce the internal bandwidth
Adaptive discrete PSO algorithm (Liu et al., 2018b)	Volume of data transfer	√	×	Decrease the number of data transmissions across data centres.

Table 1 Research on data placement strategy and scheduling policies of Hadoop workflow scheduling (continued)

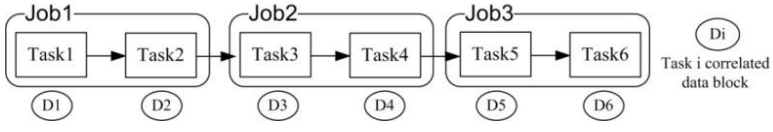
Approach	*Metrics (QoS)*	*Data placement*	*Dynamic*	*Optimisation objectives*
Hybrid GA-PSO algorithm (Manasrah and Ali, 2018)	Makespan and cost	-	×	Allocate tasks to resources efficiently
ATLAS+ (Soualhia et al., 2020)	Failure execution time	-	√	Dynamically detect the failures of the TaskTracker
FlowTime (Hu et al., 2018)	Deadlines	-	√	Simultaneously optimises the performance of ad hoc jobs
FSDP(Wang et al., 2017)	Deadlines	-	√	Achieve more reasonable fairness

3 Problem definition and statement

3.1 Cloud data-intensive SWf in Hadoop

Cloud SWf can be described as direct acyclic graphs (DAGs) $G = <V, E>$, which consist of a series of tasks (vertex) $V = <v1, v2, \ldots, vn>$ and relationships of tasks (edges) $E = <ei, j| <vi, vj> \in E>$. Each vertex *vi* represents a computing task in the workflow, and each edge *ei*, *j* indicates that *vj* depends on *vi* in the workflow. In our work, we consider only a branch data flow in a workflow since we focus on task scheduling and data placement, as shown in Figure 1. These tasks represent either map tasks or reduce tasks that perform several jobs in Hadoop. The edge relationship matrix *workflowDataTransferMap*[][] represents the data flow and dependencies between the forward task and backward task (if *workflowDataTransferMap*[*i*][*j*] > 0, then task *vi* needs to transfer intermediate data to task *vj*).

Figure 2 Cloud workflow in Hadoop



Each data block has several replicas placed on the other nodes to ensure reliability. In our workflow shown in Figure 2, *Di* represents the task-correlated data block, which is placed on three nodes according to the data block replica placement method in Hadoop (White, 2010).

The full set of notations employed in figures of this work is shown in Table 2.

Table 2 Notations

Symbol	*Notation*
Jobi	Job in workflow
Taski	Map task or reduce task
Di	Data block correlated task in Hadoop
Nodei	VM node in cloud
Racki	Rack of storage VM nodes
RiNj	Node *j* on rack *i*
ith stage	A stage including a group of nodes that can complete Taski and edges connected with nodes of previous stage in workflow
Sij	Node *j* in stage *i*, corresponds to RiNj

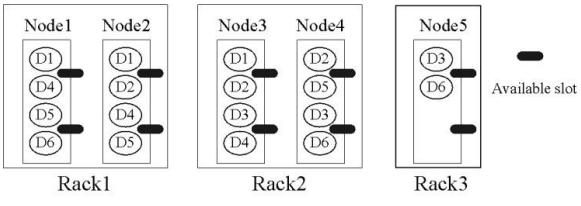
3.2 Rack-level replica placement method

In Hadoop, scheduling can be realised at the node level and rack level. Rack-level scheduling is shown in Figure 3. Data block replica placement follows the following steps:

1 The first block replica is placed on the node neighbouring the raw data block in the same rack.

2 The second replica is placed on a node in the rack neighbouring the first one.

Figure 3 Rack-level scheduling and data placement based on the data block replica placement method



3.3 Hadoop task scheduling and data locality

A task should be scheduled on a node close enough to the data block associated with the task to reduce communication latency caused by data movement. Rack-level scheduling is considered to place data blocks and replicas, as shown in Figure 3. Five nodes are distributed in the three racks. We place the data blocks and replicas of six tasks according to the replica placement method described in Section 3.2. Considering the task dependencies in the workflow, we distribute the task-correlated data blocks and the replicas on five nodes in turn for the maximum data locality.

4 The proposed method (DQ-DCWS)

The objective of the DQ-DCWS scheduling method is to minimise communication latency by ensuring the data locality and cost of cloud resources. Considering the dynamics and heterogeneity of cloud resources, a dynamic workflow scheduling method is proposed by the following steps:

Step 1 Construct a model of the optimal path for SWf.

Step 2 Establish a QoS model for the path length between two nodes in the optimal path.

Step 3 Design a DQ-DCWS algorithm based on dynamic programming.

4.1 Optimisation path model of cloud workflow

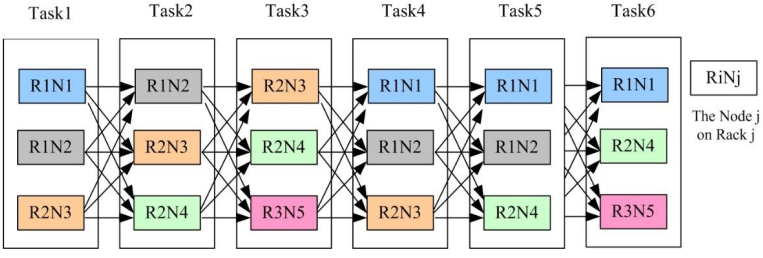
First, the scheduling process is transformed into a problem of how to choose an optimal path. As shown in Figure 4, RiNj, the j^{th} VM on the i^{th} rack, is a cloud node. To maximise data locality, the nodes that hold correlated data blocks for the same task are divided into a group according to Figure 3. For example, data block D1 which is correlated with Task1 place on three nodes (R1N1, R1N2 and R2N3). All the three nodes that are capable of performing Task1 are put into the same group. Thus, six tasks, which depend on each other, produce six groups in the workflow. Workflow scheduling selects one node from a group of nodes to execute the task. A start node, B, is added on the left, and an end node, F, is added on the right, as shown in Figure 5. The workflow scheduling problem in the work can be resolved as follows:

1 How can a node be selected from a group of nodes to execute a task?

2 Which factors determine the length of the optimal path from one node to another?

3 Which is the optimal path from B to F?

Figure 4 Optimisation path model (different colours refer to nodes from VM resources with different performance in the cloud environment) (see online version for colours)



4.2 QoS model for the length of the path between two nodes

To answer problems 1 and 2, the QoS of nodes and the communication cost between two nodes are considered to evaluate the path length of two neighbouring groups. Several QoS metrics are chosen based on our objective: VM cost, resource utilisation, response time, execution time and communication time.

The equation (1) demonstrates that the total cost of task i matching execution on a VM. The VM cost on task i include the execution cost of task i [equation (2)] and the data transfer cost from task i to other task [equation (3)].

$$VMCost[i] = TaskCost[i] + TaskDataTransferCost \tag{1}$$

$$TaskCost[i] = TaskMI[i] / vmMips * vmPerCost \tag{2}$$

$$TaskDataTransferCost[i][j] = workFlowDataTrans[i][j] * vmTransferCost[i][j] \tag{3}$$

As for the equation (2), $TaskMI[i]$ indicates runtime (mips) of task i in workflow scheduling. *vmMips* is the CPU capability of VM allocated to task i. So, $TaskMI[i]$/ *vmMips* get a cost length of task i. For each VM, a parameter *vmPerCost* denote the execution cost of VMware in unit time (mips/s) that set by cloud computing providers.

In addition to the execution cost of the task, on the workflow path, the data transfer cost of the intermediate result data which transfer to the next task after the task is completed needs to be calculated. As shown in equation (3), this value is computed by the volume of data transfer from node i to node j ($workFlowDataTrans[i][j]$) and the transfer cost between two VMs allocated to task i and task j ($vmTransferCost[i][j]$).

The costs mentioned in the above three equations refer to that users pay for scheduling their workload on the cloud VM resources (Kaur et al., 2019), including computing, storage and communication costs, computed by equation (4)

$$Cost = computing + storage + commuC \tag{4}$$

The second QoS metric is VM resource utilisation *VMRu* demonstrated in equation (5). It defines the usage of VM resources that are provided to the users for scheduling the workload [22], calculated from the ratio of *usage time* (*ut*) to *total occupancy time* (*tot*) for VM in cloud.

$$VMRu = ut / tot \tag{5}$$

The third metric VMR (VM response time) defines the time duration between the arrival of a task and the completion of a task [27] on VM nodes. Similarly, the fourth VME (VM execution time) defines the completion time that the task runs on VM nodes [18]. The last one VMC (VM communication time) defines the task communication time between two VMs. It should be noted that we consider communication time not only at the node level but also the rack level.

The above five QoS metrics are chosen to evaluate the service quality of the VM resources in the cloud. Various dimensions of the five metrics need to be normalised. Our works employ a simple (0, 1) normalisation method as equation (6), where N_{Vi} is the normalised value of the i(th) metric in above five QoS metrics of VM V, and Q_{Vi} is the QoS of the i(th) metric of VM V. $Q_i^{\min}$ is the minimum value of the i(th) metric of the VMs in a cloud data centre, and $Q_i^{\max}$ is the maximum value. These metrics include negative metrics and positive metrics, which are processed separately. For example, VM cost is a negative metric, and it is expected to obtain the minimum value.

$$N_{vi} = \begin{cases} \dfrac{Q_i^{\max} - Q_{vi}}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\min} \neq Q_i^{\max} \text{ (when } Vi \text{ is negative metric)} \\ 1 & Q_i^{\min} = Q_i^{\max} \\ \dfrac{Q_{vi} - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\min} \neq Q_i^{\max} \text{ (when } Vi \text{ is positive metric)} \end{cases} \quad (6)$$

After normalising each metric described above, the comprehensive QoS value QoSV of VM V is computed as equation (7), where W_j is the weight of the j(th) metric.

$$QoS_V = \sum_{j=1}^{5} \left(N_{V_j} \times W_j \right) \quad (7)$$

4.3 DQ-DCWS algorithm

The DQ-DCWS approach is based on dynamic programming that converts complex problems into multilevel decision issues and then recursively invokes sub problems (Fan et al., 2010). The sub problem is calculated only once, and the result will be preserved in 'memorisation'. The algorithm reuses the result of 'memorisation' so that the sacrificed space is used to improve efficiency. DQ-DCWS adopts the forward dynamic programming method.

4.3.1 Dynamic programming schema

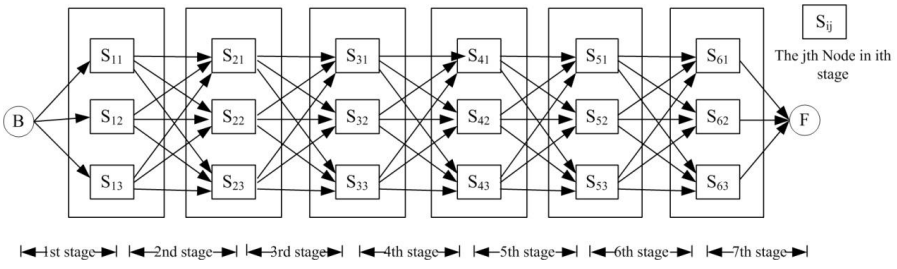
The cloud SWf is described as a DAG which begins with 'B' and ends with 'F', as shown in Figure 5, is converted from Figure 4. The DAG includes nodes and edges. For example, in $\{S_{11}, S_{12}, \ldots, S_{ij}\}$, the nodes in same group of VMs can execute a task and ensure data locality. Node S_{ij} represents the j(th) VM in the i(th) group, meaning that the i(th) task in SWf is scheduled on the j(th) VM. The edges can be represented as $\{(S_{11}, S_{21}), \ldots\ldots (S_{(i-1)(j)}, S_{(ij)})\}$.

Each group of VMs and the edges from the previous group are added to the current group to be a stage, so the DAG shown in Figure 5 obtains seven stages. When choosing a node from the next stage, the data transfer time between nodes at the rack level and the heterogeneity of cloud VM resources should be considered. In addition, the intermediate data are generated by the SWf, in which there are dependencies between tasks. The migrations of intermediate data lead to different communication costs among the nodes of the two racks.

An edge in the SWf DAG is from a VM node x_n in the n(th) stage to the i(th) node $S_{(n-1)i}$ in the $(n-1)$th stage. The value of an edge composed of QoS_{x_n} and $Comm_{s_{(n-1)i},n}$ is shown in formula (8). QoS_{x_n} quantifies the QoS of x_n according to formula (7), and $Comm_{s_{(n-1)i},n}$ is the communication cost between $S_{(n-1)i}$ and x_n, which is also quantified by formula (6).

$$Ws_{(n-1)i,^n} = QoS_{x_n} + Comm_{s_{(n-1)i},x_n} \quad (8)$$

Figure 5 DQ-DCWS approach adopts the forward dynamic programming method



4.3.2 Algorithm design

DQ-DCWS considers two main aspects: minimising the SWf cost and the execution time. First, a cost function is given by the algorithm presented in Section 4.2. The SWf DAG is initialised, including workflow tasks and VMs as the inputs. In the workflow, we consider the length of the tasks as *workflowExecutionMI* and the data transfer map relation as *workFlowDataTrans* [][]. The input data of the VM resources have three parameters: *vmMips*, which represents the computing capability of vm; *vmExecutionCost*; and *vmTransferCost*[][]. Therefore, the objective of optimisation is to maximise the length of the SWf DAG, which is formulated as (9). To schedule tasks on VMs dynamically, DQ-DCWS looks for the length $f(x_n)$ from the initial node 'B' to the current node x_n, as described in formula (9).

$$f(x_n) = \left[f\left(S_{(n-1)i}\right) + Ws_{(n-1)i,^x n} \right] \tag{9}$$

where x_n is a node in the nth stage, $S_{(n-1)i}$ is the i^{th} node in the previous $(n-1)$th stage, and $f(S_{(n-1)i})$ represents the maximum length from node 'B' to node $S_{(n-1)i}$ in the previous stage. Additionally, the communication $Ws_{(n-1)i,^x n}$ between x_n and $S_{(n-1)i}$ in $f(x_n)$ is considered.

The procedure of the DQ-DCWS algorithm based on the dynamic programming is shown in Table 3.

The DQ-DCWS algorithm includes two steps. In the OptimalWorkflow(G) step, using the QoS model and cost function described in Section 4.2, the length of the edge in the DAG with the input is calculated. The VM combination with the longest path is the VM combination that produces the maximum QoS, which refers to the efficiency of execution and the cost of workflow scheduling. The algorithm recursively calculates $f(x_n)$ one time for each node in the DAG. This calculation is performed by using memorisation to preserve the value of $f(x_n)$. The second step schedules workflow tasks along the optimal path obtained by the first step. This approach decreases the delay of scheduling tasks on heterogeneous resources.

4.4 Flowchart of DQ-DCWS

Figure 6 shows the flowchart of the optimal workflow of the DQ-DCWS approach. The procedure begins with B and recursively computes the length of path (n) to achieve the maximum length in each stage until node F is reached. The longest path (n) and output memory are employed in the workflow scheduling.

Table 3 DQ-DCWS algorithm

Algorithm DQ-DCWS

1 OptimalWorkflow(G):

Input: an Workflow DAG G (n tasks, m VMs), a set of available VM nodes S, the weight of the edge from node i to node j W_{ij}, Beginning node B, Finish node F,

Output: VM list in the longest path from B to F vms_optimal[], the optimal path from x_n to the finish node $f(x_n)$, the previous node of x_n in the $(n-1)$th stage on the optimal path $P(x_n)$

x_n: one node of stage n

Sn_i: VMware node i in stage n

memorisation(n)(i): (the length of the optimal path from node S_{ni} to finish node)

1: Set $x_n = B$

2: **for each** stage n in G **do**

3: **if** memorisation(n)(x_n) !=0 **then**

4: $f(x_n) \leftarrow$ memorisation(n)(x_n)

5: **else**

6: **for** each S in stage $(n-1)$ **do**

7: select $S_{(n-1)i}$ from $\{S_{(n-1)1}, S_{(n-1)2}, \ldots S_{(n-1)m}\}$

$$s.t: \left[f\left(S_{(n-1)i}\right) + Ws_{(n-1)i,^x n}\right] == \max\left[f\left(x_{(n-1)}\right) + Wx_{(n-1),^x n}\right]$$

8: $$f\left(x_n\right) \leftarrow \left[f\left(S_{(n-1)i}\right) + Ws_{(n-1)i,^x n}\right]$$

9: $P(x_n) \leftarrow S_{(n-1)i}$ and memorisation(n)(x_n) $\leftarrow f(x_n)$

10: vms_optimal[$n-1$]= $S_{(n-1)i}$

11: **end if**

12: **end for**

13: **return** vms_optimal[]

2 Cloud workflow scheduling:

Input: W_{ij}, $P(x_n)$, $f(x_n)$, task list task[], vms_optimal[]

Output: Scheduling decision $SD = \{(task[n] <-> b_n) | b_n$ is a node of n stage on optimal path scheduling decision$\}$

S_{nj}: the vm node j on optimal path in stage n from vms_optimal[]

1: Set $x_n = B$

2: **for each** stage n in G **do**

3: if (the VM S_{nj} cannot execute the task) then

4: $a \leftarrow P(S_{nj})$

5: Find a vm node b_n in stage n st:

6: $$\left[W_{ab_n} + f\left(b_n\right)\right] == \max\left[W_{ak} + f(k)\right]$$ (k is the node other than S_{nj} in stage n)

7: else

8: $b_n \leftarrow S_{nj}$

9: Allocate available VM resource b_n to task[n]

10: **end if**

11: **end for**

Figure 6 Optimal workflow flowchart

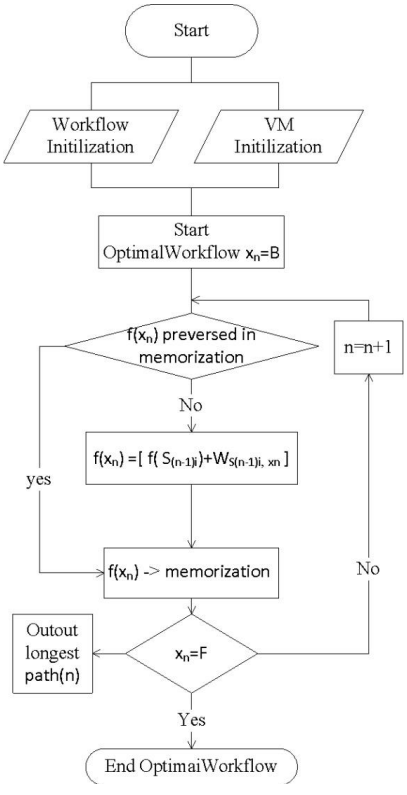
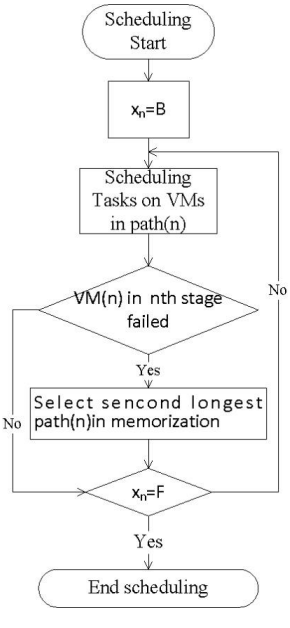


Figure 7 Scheduling flowchart



The scheduling flowchart is shown in Figure 7. It is obvious that the simple method enables a fast response to a single point of failure and obtains the second longest path (n).

5 Analysis

In this section, we analyse the performance of the proposed DQ-DCWS method for a real Hadoop workload in a heterogeneous and scalable environment. The result of this analysis is compared with three general workflow scheduling algorithms. Our experiments verify the performance of the algorithm from two aspects of running time and cost.

5.1 Application

A real-life SWf Montage (Deelman et al., 2008) is used in our experiment. Montage is a data-intensive application for assembling flexible image transport system (FITS) images into custom mosaics, and it was created by the NASA/IPAC Infrared Science Archive. The Montage was chosen as a respectively workflow in our work because we focus on reducing data-block movement in data-locality which involve the amount of time the application spends waiting for I/O and Montage have a significant advantage for I/O-bound. In our experiments we configured Montage workflow to contain 35 jobs (35 montage executable core modules). The size of a Montage workflow is based on the area of the sky covered by the output mosaic (NASA, 2010). The Montage workflow application job is created with ten tasks, which generate 1-degree mosaics of the sky using a 1.78 MB image (portion of galactic plane) as input data.

5.2 Experimental environment

We ran experiments on Amazon EC2 which was chosen because it is a popular and feature-rich cloud resource. Real Hadoop clusters were deployed on Amazons EC2 with varying number of VM instances from 20 to 100. EC2 VMs adopt the most cost-effective processor c1.medium (Lustre, 2015).

Table 4 Hadoop clusters VMs parameters

Parameter	*Value*
Number of VMs	20–100
HOST_PEs	2
MIPS	1,000
RAM	2,048 (*MB*)
Storage	Local
BW	10,000 (*mbps*)
Price of computing resources	\$0.20/hr
Chargers of VMs	\$0.15 GB/Month

A series of experiments performed the Montage workflow on the configured Hadoop clusters using the parameters presented in Table 4. These parameters are used to identify

the capability of the VMs and the resource cost of the cloud system. The program runs on different numbers of VMs and is applied to achieve two objectives:

1 Optimising the workflow execution time to reduce the makespan which means the total program runtime.

2 Reducing the task execution cost.

5.3 Algorithm comparison

To evaluate the performance of the proposed algorithm, the obtained results of our algorithm are compared with four existing scheduling algorithms: FairScheduler in Hadoop MapReduce, PSO, IPSO and FPSO.

FairScheduler: one of the default schedulers in Hadoop that is unable to implement data locality. FairScheduler assigns resources to jobs, and all jobs receive an equal share of resources over time. FairScheduler organises the jobs into pools and divides resources fairly between these pools (APACHE, 2022).

- PSO: recently, PSO has been used to optimise the cloud scheduling problem, and a number of particles (equal to the solutions for the application problem) represent a swarm moving around to search for the best solution in the search space. An application has a fitness function to evaluate the fitness value by particles moving at certain velocities. PSO can obtain fast convergence, but the whole swarm is easily trapped in a local optimum.
- IPSO: some works have improved the traditional PSO for SWf application scheduling in the scientific field (Rashmi and Basu, 2017), such as the previous work (Chen and Zhang, 2012), which proposed a new PSO called IPSO that records not only the best position but also the worst position to avoid being trapped in a local optimum.
- FPSO: both PSO and IPSO are only scheduling optimisations and did not consider data placement based on data locality. Hence, a recent approach is also compared: FPSO strategy (Kchaou et al., 2022). The FPSO combine the IT2DFCM strategy for data placement with the PSO technique for task scheduling to execute SWf. The objective of the FPSP is to minimise the overall data movement number.

In our comparison, the swarm parameters applied in PSO, IPSO and FPSO are listed in Table 5.

Table 5 Swarm algorithm parameters

Parameter	*Value*
Global increment	0.9
Inertia	0.95
Number of particles	25
Partial increment	0.9
Velocity graph factor	10
Evolve	10

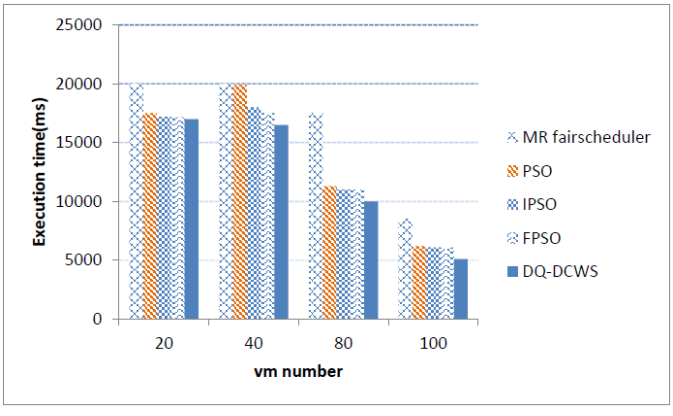
5.4 Experimental analysis

Experiments are performed on four different numbers of VMs (20, 40, 80, and 100 VMs) and are repeated 100 times for each different number of VMs. The average results of the 100 replicates are compared to those of the other four algorithms. To evaluate the performance of these five approaches, two performance metrics are applied: SWf execution time and task running cost.

Execution time represents the makespan from workflow submission to task completion. From Figure 8, DQ-DCWS reduces the execution time well and is significantly better than FairScheduler in Hadoop MapReduce. Since FairScheduler is static and does not have data locality, it spends some time on data movement and recovering a single failure. As shown in Table 6, DQ-DCWS achieves a 26.5% improvement compared to FairScheduler. DQ-DCWS performs better than PSO, IPSO and FPSO, improving by 11.5%, 7.1% and 6.1 %, respectively. The advantage of DQ-DCWS is that it can dynamically schedule tasks on VMs and cannot be trapped in a local optimum. Detailed data are provided in Table 6.

To observe the cloud cost-efficient performance, the execution cost on cloud resources is qualified and evaluated. Figure 9 shows the experimental results of the five approaches. The cost efficiency of DQ-DCWS is better than that of the other four approaches. In small-scale VMs, such as 20 and 40, the DQ-DCWS cost is still close to that of the others. However, as the number of VMs increases, the superiority of DQ-DCWS becomes increasingly obvious. As shown in Table 6, cost reductions of 12.5%, 11.8% and 11.6% are achieved by comparing with FairScheduler, PSO and IPSO respectively. On the recent method FPSO, our approach also obtains a 7.0% improvement in cost. Detailed data are provided in Table 6.

Figure 8 Execution time of SWf with ten tasks executed by five algorithms on different numbers of VMs (see online version for colours)



These reasons lead to the enhancement:

1 In the running time of the experiments, FairScheduler spend amount of time on movement of data block which cannot satisfy data-locality when tasks are scheduled to data nodes.

2 PSO and IPSO require plenty of time to search for the optimal practicality in the whole swarm every time they are run. The large amount of data movement in task execution is the main reason for the increased cost of these three strategies.

3 Although FPSO policy reduces data movement by the data placement procedure, workflow scheduling is still static and fails to search for the best VM to perform tasks.

The proposed DQ-DCWS always chooses the correct VMs, not fast VMs, to run tasks. The proposed approach is devoted to obtaining data locality and cost efficiency and to finding the optimal solution faster than other algorithms.

Figure 9 Execution cost of SWf for ten tasks executed by five algorithms on different numbers of VMs (see online version for colours)

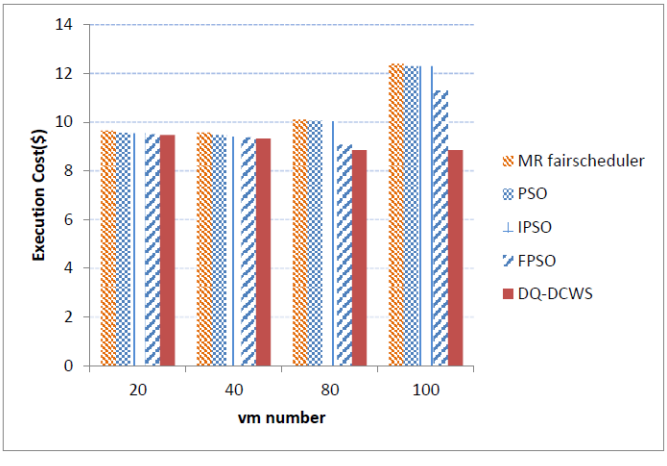


Table 6 Comparison of DQ-DCWS with the other approaches in terms of three metrics ([(other approaches)-DQ-DCWS]/(other approaches) × 100

Performance metrics of improvement percentage	*MR FairScheduler (%)*	*PSO (%)*	*IPSO (%)*	*FPSO (%)*
Execution time	26.5	11.5	7.1	6.1
Cost improvement	12.5	11.8	11.6	7.0

6 Conclusions

As the complexity of data-intensive applications has increased, SWf has emerged as an important technology. In our work, we propose a novel DQ-DCWS approach for SWf scheduling. Based on data locality, the efforts to improve the execution efficiency and cost efficiency are validated when running data-intensive SWf on heterogeneous cloud resources.

The SWf scheduling problem is modelled as an optimisation path issue. Five metrics of QoS are selected to evaluate the length of the SWf path. A dynamic programming method is employed to design an algorithm to choose an optimal path. The superiority of

this algorithm's performance over that of four other workflow scheduling methods is demonstrated through real-life experiments. The proposed approach is better than the others, with high efficiency for makespan and the cost of VM resources. In future work, this approach can be applied to other real-life applications in big data architectures.

References

Afgan, E., Baker, D., Batut, B., Van den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B., Guerler, A., Hillman-Jackson, J., Jalili, V., Rasche, H., Soranzo, N., Goecks, J., Taylor, J., Nekrutenko, A. and Blankenberg, D. (2018) 'The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update', *Nucleic Acids Research*, Vol. 46, No. W1, pp.W537–W544.

Alwidian, J.A. and Alaa Abdallat, A.A. (2019) 'Hadoop MapReduce job scheduling algorithms survey and use cases', *Modern Applied Science*, Vol. 13, No. 7, pp.38–52.

APACHE (2022) *Fair Scheduler* [online] https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html (accessed 29 September 2022).

Amazon.com (2022) *Elastic Compute Cloud (EC2)* [online] http://aws.amazon.com/ec2 (accessed 27 October 2022).

Arabnejad, H. and Barbosa, J.G. (2017) 'Multi-QoS constrained and profit-aware scheduling approach for concurrent workflows on heterogeneous systems', *Future Generation Computer Systems*, Vol. 68, No. 3, pp.211–221.

Bae, M., Yeo, S., Park, G. and Oh, S. (2020) 'Novel data-placement scheme for improving the data locality of Hadoop in heterogeneous environments: NA', *Concurrency & Computation Practice & Experience*, Vol. 33, No. 18, p.e5752.

Borthakur, D. (2008) *HDFS Architecture Guide*, HADOOP APACHE PROJECT [online] http://hadoop.apache.org/common/docs/current/hdfsdesign.pdf.

Chen, W.N. and Zhang, J. (2012) 'A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints', *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

Choi, D., Jeon, M., Kim, N. and Lee, B-D. (2018) 'An enhanced data-locality-aware task scheduling algorithm for Hadoop applications', *IEEE Systems Journal*, Vol. 12, No. 4, pp.3346–3357.

Davis, D.A.P., Plante, J.M., Chaniotakis, E., Guok, C. and Vokkarane, V.M. (2017) 'Enhancing ESnet's OSCARS path computation engine', *GLOBECOM 2017 – 2017 IEEE Global Communications Conference*.

Dean, J. and Ghemawat, S. (2004) 'MapReduce: simplified data processing on large clusters', *Proceedings of Sixth Symposium on Operating System Design and Implementation (OSD2004)*.

Deelman, E., Ferreira Da Silva, R., Vahi, K., Rynge, M., Mayani, R., Tanaka, R., Whitcup, W. and Livny, M. (2021) 'The Pegasus workflow management system: translational computer science in practice', *Journal of Computational Science*, Vol. 52, in press.

Deelman, E., Singh, G., Livny, M., Berriman, B. and Good, J. (2008) 'The cost of doing science on the cloud: the Montage example', *ACM/IEEE Conference on High Performance Computing*, Austin, Texas, USA, 15–21 November.

Dey, N. and Gunasekhar, T. (2019) 'A comprehensive survey of load balancing strategies using Hadoop queue scheduling and virtual machine migration', *IEEE Access*, Vol. 7, No. 13, pp.92259–92284.

Fan, D., Zhang, R., Ruan, K., Lin, J. and Zhao, Z. (2010) 'A QoS-based scheduling approach for complex workflow applications', *2010 Fifth Annual ChinaGrid Conference*, IEEE, Guangzhou, China.

Fang, B. and Sun, L. (2018) 'Cloud workflow scheduling optimization oriented to QoS and cost-awareness', *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems CIMS*, Vol. 24, No. 2, pp.331–348.

Gandomi, A., Reshadi, M., Movaghar, A. and Khademzadeh, A. (2019) 'HybSMRP: a hybrid scheduling algorithm in Had MapReduce framework', *Journal of Big Data*, Vol. 6, No. 10, p.106.

Garg, R. and Singh, A.K. (2014) 'Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization', *Journal of Supercomputing*, Vol. 68, No. 2, pp.709–732.

Guo, Z., Fox, G. and Zhou, M. (2012) 'Investigation of data locality in MapReduce', *IEEE/ACM International Symposium on Cluster Cloud & Grid Computing*, pp.419–426.

Hu, C., Min, W., Liu, G. and Wen, X. (2007) 'QoS scheduling algorithm based on hybrid particle swarm optimization strategy for grid workflow', *Sixth international Conference on Grid and Cooperative Computing*, *GCC 2007*.

Hu, Z., Li, B., Chen, C. and Ke, X. (2018) 'FlowTime: dynamic scheduling of deadline-aware workflows and ad-hoc jobs', *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, pp.929–938.

Javanmardi, A.K., Yaghoubyan, S.H., Bagherifard, K., Nejatian, S. and Parvin, H. (2021) 'A unit-based, cost-efficient scheduler for heterogeneous Hadoop systems', *Journal of Supercomputing*, Vol. 77, No. 1, pp.1–22.

Kalia, K., Dixit, S., Kumar, K., Gera, R., Epifantsev, K., John, V. and Taskaeva, N. (2022) 'Improving MapReduce heterogeneous performance using KNN fair share scheduling', *Robotics and Autonomous Systems*, Vol. 157, No. 12, pp.22–35.

Kasztelnik, M., Coto, E., Bubak, M., Malawski, M., Nowakowski, P., Arenas, J., Saglimbeni, A., Testi, D. and Frangi, A.F. (2017) 'Support for Taverna workflows in the VPH-share cloud platform', *Computer Methods and Programs in Biomedicine*, Vol. 146, No. 2, pp.37–46.

Kaur, S., Bagga, P., Hans, R. and Kaur, H. (2019) 'Quality of service (QoS) aware workflow scheduling (WFS) in cloud computing: a systematic review', *Arabian Journal for Science and Engineering*, Vol. 44, No. 4, pp.2867–2897.

Kchaou, H., Kechaou, Z. and Alimi, A.M. (2022) 'A PSO task scheduling and IT2FCM fuzzy data placement strategy for scientific cloud workflows', *Journal of Computational Science*, Vol. 64, No. 2022, pp.1–13.

Kennedy, J.R.E. (1995) 'Particle swarm optimization', *IEEE International Conference on Neural Networks*, *Proceedings of ICNN'95 1942–1948*.

Kim, H. and Kim, Y. (2018) 'An adaptive data placement strategy in scientific workflows over cloud computing environments', *NOMS IEEE/IFIP Network Operations & Management Symposium*.

Li, X., Jia, X. and Yun, Y. (2015) 'A chaotic particle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems', *Computational Intelligence & Neuroscience*, Vol. 2015, No. 3, pp.1–11.

Liu, Y., Wu, C.Q., Wang, M., Hou, A. and Wang, Y. (2018a) 'On a dynamic data placement strategy for heterogeneous Hadoop clusters', *2018 International Symposium on Networks, Computers and Communications (ISNCC)*.

Liu, Z., Xiang, T., Lin, B., Ye, X., Wang, H., Zhang, Y. and Chen, X. (2018b) 'A data placement strategy for scientific workflow in hybrid cloud', *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp.556–563.

Lumley, J.A., Sharman, G., Wilkin, T., Hirst, M., Cobas, C. and Goebel, M. (2020) 'A KNIME workflow for automated structure verification', *SLAS Discovery*, Vol. 25, No. 8, pp.950–956.

Lustre, A. (2015) *Running Scientific Workflow Applications on the Amazon EC2 Cloud* [online] https://irsa.ipac.caltech.edu/Montage/publications/AstroinformaticsPoster2_final.pdf (accessed 9 September 2022).

Manasrah, A.M. and Ali, H.B. (2018) 'Workflow scheduling using hybrid GA-PSO algorithm in cloud computing', *Wireless Communications and Mobile Computing*, Vol. 2018, No. 1, p.16.

Mon, E.E., Thein, M.M. and Aung, M.T. (2017) 'Clustering based on task dependency for data-intensive workflow scheduling optimization', *Workshop on Many-Task Computing on Clouds*.

Mondelli, M.L.B., Magalhães, T.T., Loss, G., Wilde, M. and Gadelha, L.M.R. (2018) 'BioWorkbench: a high-performance framework for managing and analyzing bioinformatics experiments', *Peer J.*, Vol. 6, No. 1, p.e5551.

NASA (2010) *Applications of Montage – Image Gallery* [online] http://vaoweb3.ipac.caltech.edu/gallery.html (accessed 15 March 2022).

Nasr, A., El-Bahnasawy, N., Attiya, G. and El-Sayed, A. (2018) 'Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint', *Arabian Journal for Science and Engineering*, Vol. 44, No. 4, pp.3765–3780.

Owsiak, M., Plociennik, M., Palak, B., Zok, T., Reux, C., Gallo, L.D., Kalupin, D., Johnson, T. and Schneider, M. (2017) 'Running simultaneous Kepler sessions for the parallelization of parametric scans and optimization studies applied to complex workflows', *Journal of Computational Science*, Vol. 20, No. 2017, pp.103–111.

Qian, S., Romanus, M., Tong, J., Yu, H., Bremer, P.T., Petruzza, S., Klasky, S. and Parashar, M. (2017) 'In-staging data placement for asynchronous coupling of task-based scientific workflows', *International Workshop on Extreme Scale Programming Models & Middlewar*.

Qureshi, B. (2019) 'Profile-based power-aware workflow scheduling framework for energy-efficient data centers', *Future Generation Computer Systems – The International Journal of eScience*, Vol. 94, No. 2019, pp.453–467.

Rashmi, S. and Basu, A. (2016) 'A. deadline constrained cost effective workflow scheduler for Hadoop clusters in cloud datacenter', *International Conference on Computation System & Information Technology for Sustainable Solutions*.

Rashmi, S. and Basu, A. (2017) 'Resource optimised workflow scheduling in Hadoop using stochastic hill climbing technique', *IET Software*, Vol. 11, No. 5, pp.239–244.

Reddy, K.H.K. and Roy, D.S. (2018) 'DPPACS: a novel data partitioning and placement aware computation scheduling scheme for data-intensive cloud applications', *Computer Journal*, Vol. 59, No. 1, pp.64–82.

Reddy, K.H.K., Pandey, V. and Roy, D.S. (2019) 'A novel entropy-based dynamic data placement strategy for data intensive applications in Hadoop clusters', *International Journal of Big Data Intelligence*, Vol. 6, No. 3, pp.20–37.

Seethalakshmi, V., Govindasamy, V. and Akila, V. (2020) 'Real-coded multi-objective genetic algorithm with effective queuing model for efficient job scheduling in heterogeneous Hadoop environment', *Journal of King Saud University – Computer and Information Sciences*, Vol. 34, No. 6, pp.3178–3190.

Service, F.R. (2012) 'Materials scientists look to a data-intensive future', *Science*, Vol. 335, No. 6075, pp.1434–1435.

Seth, S. and Singh, N. (2020) 'Multi-QoS enhanced heuristic model for scheduling of scientific workflows', *International Journal of Entific & Technology Research*, Vol. 9, No. 33, pp.202–210.

Singh, P.V., Kaur, A. and Singh, M. (2018) 'Heuristic data placement and replication for scientific workflow in cloud computing', *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*.

Soualhia, M., Khomh, F. and Tahar, S. (2020) 'A dynamic and failure-aware task scheduling framework for Hadoop', *IEEE Transactions on Cloud Computing*, Vol. 8, No. 2, pp.553–569.

Thingom, C., Ganesh, K.R. and Yeon, G. (2018) 'Workflow scheduling using heuristic scheduling in Hadoop', *Journal of Information & Communication Convergence Engineering*, Vol. 16, No. 4, pp.264–270.

Tong, S. and Wu, C.Q. (2017) 'Performance optimization of Hadoop workflows in public clouds through adaptive task partitioning', *IEEE Infocom – IEEE Conference on Computer Communications*.

Vengadeswaran, S. and Balasundaram, S.R. (2019) 'CORE – an optimal data placement strategy in Hadoop for data intensive applications based on cohesion relation', *International Journal of Computer Systems Science & Engineering*, Vol. 34, No. 1, pp.47–59.

Verma, A. and Kaushal, S. (2014) 'Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud', *Engineering & Computational Sciences*.

Wang, J., Shang, P. and Yin, J. (2013) 'DRAW: a new Data-gRouping-AWare data placement scheme for data intensive applications with interest locality', *IEEE Transactions on Magnetics*, Vol. 49, No. 6, pp.2514–2520.

Wang, Y., Cao, S., Guan, W., Zhen, F. and He, G. (2017) 'Fairness scheduling with dynamic priority for multi workflow on heterogeneous systems', *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*.

White, T. (2010) *Hadoop: The Definitive Guide*, O'Reilly Media.

Wu, J.X., Zhang, C.S., Zhang, B. and Wang, P. (2016) 'A new data-grouping-aware dynamic data placement method that take into account jobs execute frequency for Hadoop', *Microprocessors & Microsystems*, Vol. 47, No. 1, pp.161–169.

Wu, Z., Ni, Z., Gu, L. and Xiao, L. (2011) 'A revised discrete particle swarm optimization for cloud workflow scheduling', *International Conference on Computational Intelligence & Security*.

Wylie, A., Shi, W., Corriveau, J.P. and Wang, Y. (2016) 'A scheduling algorithm for Hadoop MapReduce workflows with budget constraints in the heterogeneous cloud', *IEEE International Parallel & Distributed Processing Symposium Workshops*.

Xu, J. and Yong, T. (2015) 'Improved particle optimization algorithm solving Hadoop task scheduling problem', *2nd International Conference on Intelligent Computing and Cognitive Informatics (ICICCI 2015)*.

Ye, Q., Wu, C.Q., Cao, H., Rao, N.S.V. and Hou, A. (2018) 'Storage-aware task scheduling for performance optimization of big data workflows', *16th IEEE ISPA/17th IEEE IUCC/8th IEEE BDCloud/11th IEEE SocialCom/8th IEEE SustainCom*.

Zhang, J., Tan, W., Alexander, J., Foster, I. and Madduri, R. (2011) 'Recommend-as-you-go: a novel approach supporting services-oriented scientific workflow reuse', *2011 IEEE International Conference on Services Computing (SCC)*.

Zhang, Q., Zhani, M.F., Yang, Y., Boutaba, R. and Wong, B. (2017) 'PRISM: fine-grained resource-aware scheduling for MapReduce', *IEEE Transactions on Cloud Computing*, Vol. 3, No. 2, pp.182–194.

Zheng, H., Dongjin, Y.U., Zhang, L., Computer, S.O. and University, H.D. (2017) 'Multi-QoS cloud workflow scheduling based on firefly algorithm and dynamic priorities', *Computer Integrated Manufacturing Systems*, Vol. 23, No. 5, pp.963–971.

Zhou, C.J., Zong, P. and Computer, S.O. (2019) *Improvement of Backup Data Placement Policy of Hadoop*, Nanjing University of Posts and Telecommunications, PhD.