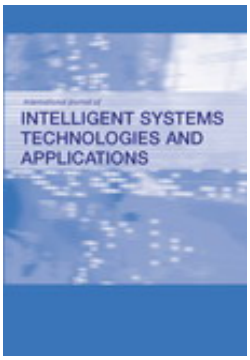


International Journal of Intelligent Systems Technologies and Applications

ISSN online: 1740-8873 - ISSN print: 1740-8865
https://www.inderscience.com/ijista

Service capability aware big data workflow scheduling approach in cloud datacentre

Jie Cao, Jinchao Xu, Bo Wang

DOI: 10.1504/IJISTA.2024.10059902

Article History:

Received:	24 July 2022
Last revised:	13 December 2022
Accepted:	22 August 2023
Published online:	05 February 2024

Service capability aware big data workflow scheduling approach in cloud datacentre

Jie Cao

Software Engineering College,
Zhengzhou University of Light Industry,
Zhengzhou, 450002, China
Email: 42675492@qq.com

Jinchao Xu*

Information Centre,
Shanghai Jiaotong University,
Shanghai, 200240, China
Email: xujc@sjtu.edu.cn
*Corresponding author

Bo Wang

Software Engineering College,
Zhengzhou University of Light Industry,
Zhengzhou, 450002, China
Email: 735693062@qq.com

Abstract: With the increasing application of cloud computing, big data workflow scheduling in cloud datacentre also become an important focus of research. How to guarantee minimal scheduling length is the main challenge in scheduling workflow in cloud-based environments. The main limitation of proposed approaches stems is that they overlook the service capability support levels of the virtual machines and service capability requirement levels of the different tasks in a workflow, thus risking resulting in extremely poor processing efficiency. We propose a service dynamic level scheduling algorithm in cloud datacentre (Cloud-SDLS) that consists of three stages: virtual machines' service capability support computation, tasks' service capability requirement computation, and service dynamic level scheduling. Experimental results show that the proposed algorithms effectively satisfy the QoS in service capability requirement. It is significant to shorten workflow completion time in practice.

Keywords: cloud computing; service capability requirement; service capability support; workflow scheduling.

Reference to this paper should be made as follows: Cao, J., Xu, J. and Wang, B. (2024) 'Service capability aware big data workflow scheduling approach in cloud datacentre', *Int. J. Intelligent Systems Technologies and Applications*, Vol. 22, No. 1, pp.1–15.

Biographical notes: Jie Cao received his MS in Computer Technology from the Shanghai Maritime University in 2010, and PhD in Computer Science from the Tongji University, Shanghai, China, in 2015. He is currently a Lecturer at the Software Engineering College, Zhengzhou University of Light Industry. His research interests include distributed systems, cloud computing, resource management and big data computing platform.

Jinchao Xu received his MS in Mathematics from the Shandong Normal University in 2008, and PhD in Computer Science from the Tongji University in 2013. He is currently a Lecturer at the School of Medicine, Shanghai Jiaotong University. His research interests include software security, cloud computing and artificial intelligence.

Bo Wang received his PhD in Computer Science from the Xi'an Jiaotong University, Xian, China in 2017. He is currently a Lecturer at the Software Engineering College, Zhengzhou University of Light Industry. His research interests include distributed systems, cloud computing, resource management and big data computing platform.

1 Introduction

Big data workflows processing has become crucial due to massive analytics demands in all the major business and scientific domains such as banking, fraud detection, healthcare, demand forecasting, and scientific explorations (Smith, 2020). Cloud computing play a very important role in big data workflows processing. However, scheduling big data workflows to VM clusters so that they can be processed efficiently is a very difficult task. In addition, due to the diversity of VM types provided by cloud service providers, the performance of VMs varies greatly, and the effect of processing different types of tasks varies greatly, which further increases the difficulty of scheduling big data workflows. It became a serious problem to efficiently schedule big data tasks to cloud datacentre to guarantee better performance (Rjoub et al., 2020).

Researchers have proposed many classical algorithms for workflow scheduling in distributed computing systems. Such as HEFT (Topcuouglu et al., 2002), Min-Min (Blythe et al., 2005), and Max-Min (Braun et al., 2001). In recent years, more algorithms on considering availability and cost are proposed. Xu et al. propose an algorithm of task non-replication for minimising resource consumption with a reliability goal (Xu et al., 2018). Chen and Xiao (2019) present the search and earliest finish time (SEFT) algorithm for a bounded number of heterogeneous computing nodes intending to minimise the computing cost for scheduling DAGs on the heterogeneous cloud platform. Wei et al. (2018) propose a variety schedule algorithms for big data processing workflow applications on clouds to minimise the cost and satisfy the deadline constraints. Wang et al. (2012) propose a trust-based dynamic level scheduling algorithm called Cloud-DLS to minimise time costs and ensure a secure execution of tasks. It is a Bayesian approach for a cognitive trust model which relies on direct and indirect trust sources to derive trust values for cloud resources. Sun et al. (2015) proposed re-stream, an energy-efficient scheduling algorithm to optimise the energy consumption and response time for executing workflow applications. Ghafarian and Javadi (2015) propose a workflow scheduling system that partitions a workflow into sub-workflows to minimise data

dependencies among the sub-workflows. Chen et al. (2020) propose a novel real-time scheduling algorithm using task-duplication to minimising both the completion time and monetary cost of processing big data workflows in clouds. Dai et al. (2018) propose a resource allocation algorithm to make trade-offs among the performance, availability and cost for processing big data applications on clouds. In Wang et al. (2017), the authors propose a data-dependency-driven task scheduling scheme, named D3S2, for big data processing. D3S2 is mainly composed of two parts: dependency-aware placement mechanism, and transfer-aware task scheduling mechanism.

Focus on the workflow scheduling problem and consider the service capability support levels of the virtual machines and service capability requirement levels of the different tasks in a workflow, in this paper, we propose an efficient workflow scheduling algorithm called the service dynamic level scheduling algorithm in cloud datacentre (Cloud-SDLS), considering VMs' service capability support and tasks' service capability requirement.

2 Problem description

2.1 Modelling big data workflow

Definition 1 (Big data workflow): A certain big data workflow in cloud environments is a directed acyclic graph (DAG), which is described as $G_i = (V_i, E_i, Q_i, D_i, O_i, C_i)$.

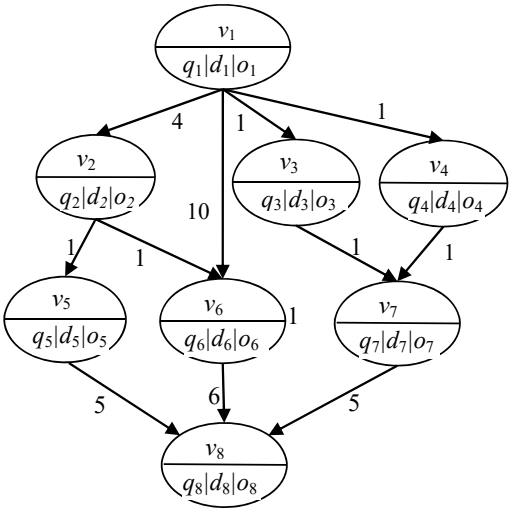
- $V_i = \left\{v_1^i, v_2^i, \cdots, v_{|V_i|}^i\right\}$: the task set in the workflow G_i, v_j^i is the j^{th} task in the workflow G_i.
- $E_i \subseteq V_i \times V_i$: the set of directed edges between tasks, edge $e_{p,q}^i \in E_i$ represents the precedence constraint that task v_p^i should be finished before the task v_q^i starts, here v_p^i is a direct precursor of v_q^i and v_q^i is a direct successor of v_p^i.

  $pred(v_j^i)$ denotes the set of all the direct precursors of v_j^i and $succ(v_j^i)$ describes the set of all the direct successors of v_j^i.
- $Q = \left\{q_1^i, q_2^i, \cdots, q_{|V_i|}^i\right\}$ is the set of the computational workload in some machine instructions for V_i, and a q_j^i denotes the computational workload for task v_j^i.
- $D = \left\{d_1^i, d_2^i, \cdots, d_{|V_i|}^i\right\}$ is the set of the data process workload for V_i, and a d_j^i denotes the data process workload for task v_j^i.
- $O = \left\{o_1^i, o_2^i, \cdots, o_{|V_i|}^i\right\}$ is the set of the I/O workload for V_i, and a o_j^i denotes the I/O workload for task v_j^i. The data to be communicated between tasks can be transmitted only after I /O processing, such as coding and encryption before data transmission.
- $C = \left\{c_{k,j}^i \middle| v_k^i, v_j^i \in V_i\right\} \subseteq V_i \times V_i$ is the data transfer workload between tasks.

Besides, in a given workflow graph, if $pred(v_j^i) = \phi$, we call v_j^i entry task, if $succ(v_j^i) = \phi$, we call v_j^i exit task. We assume that a workflow has exactly one entry task and one exit task.

A sample workflow denoted as w is shown in Figure 1. The eight tasks of a workflow described as $\{v_1, v_2, \cdots, v_8\}$. The associated weights on the edges denote the data transfer workload between the tasks.

Figure 1 A sample workflow



2.2 Modelling cloud datacentre

We assume that the cloud datacentre consists of different types of virtual machines offered by IaaS cloud service providers. Assume that the average bandwidth between virtual machines is roughly equal, different types of virtual machines have different resource configurations, such as different networks, CPU and memory resources, etc.

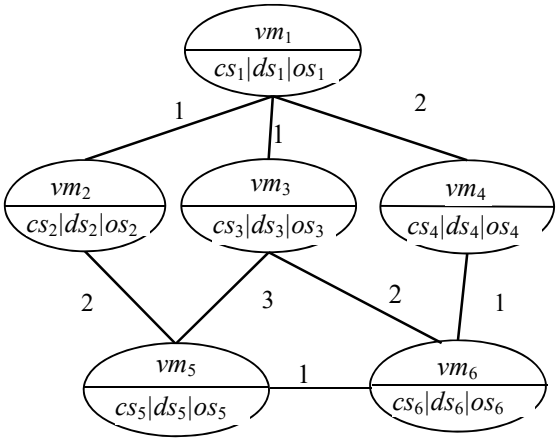
Definition 2 (Cloud datacentre): A specific cloud data centre can be represented as an undirected graph, which is described as *Cloud* = (*VM*, *CS*, *DS*, *OS*, *L*, *B*),

- $VM = \{vm_1, vm_2, \ldots, vm_m\}$ is the VM set in *Cloud*, and vm_i denotes the i^{th} VM in cloud.
- $CS = \{cs_1, cs_2, \ldots, cs_m\}$ is the set of the calculating speed of VMs, and cs_i denotes the calculating speed of vm_i.
- $DS = \{ds_1, ds_2, \ldots, ds_m\}$ is the set of the data process speed of VMs, and ds_i denotes the data process speed of vm_i.
- $OS = \{os_1, os_2, \ldots, os_m\}$ is the set of the I/O process speed of VMs, and os_i denotes the I/O process speed of vm_i.
- $L = \{l_{ij}\}$ is the set of communication links between VMs.

- $B = \{b_{ij}\}$ is the set of communication bandwidth of the links in L. b_{ij} is the data volume go through l_{ij} at certain intervals.

A sample cloud datacentre denoted as *cloud* is shown in Figure 2. The six VMs in *cloud* are described as $VM = \{vm_1, vm_2, \ldots, vm_6\}$. The associated weights on the edges denote the communication bandwidth between the VMs.

Figure 2 A sample cloud datacentre



2.3 Problem formulations

Let $EST(v_j^i, vm_m)$ be the earliest execution start time of v_j^i on vm_m, and $EFT(v_j^i, vm_m)$ be the earliest finish time v_j^i on vm_m.

$EST(v_j^i, vm_m)$ and $EFT(v_j^i, vm_m)$ can be computed by (1) and (2), respectively.

$$EST\left(v_j^i, vm_m\right) = \max\left\{Available\left(v_j^i, vm_m\right), \max_{v_k^i \in pred\left(v_j^i\right)}\left\{EFT\left(v_k^i, vm_n\right) + ct_{k,j}^i\right\}\right\} \quad (1)$$

$$EFT\left(v_j^i, vm_m\right) = EST\left(v_j^i, vm_m\right) + \frac{q_j^i}{cs_m} + \frac{d_j^i}{ds_m} + \frac{o_j^i}{os_m} \quad (2)$$

where $ct_{k,j}^i = \frac{ct_{k,j}^i}{b_{mn}}$, is the communication time of edge $e_{k,j}^i$ transferring data from task v_k^i (scheduled on vm_n) to task v_j^i (scheduled on vm_m).

For the entry task v_{entry}^i, $EST(v_{entry}^i, vm_m) = 0$.

For the exit task v_{exit}^i, $EFT(v_{exit}^i)$ is the whole schedule length. One of the goals of our paper is to minimise $EFT(v_{exit}^i)$.

2.4 Service capability

A cloud datacentre consisting of different VMs is often referred to as a heterogeneous cloud computing system. There may be multiple scheduling schemes for a big data workflow to be executed on VMs of heterogeneous cloud computing systems, and different scheduling schemes will produce different execution effects. The execution effect of a big data workflow on heterogeneous cloud computing systems depends on the computational speed of VMs, data processing speed, communication data processing speed, transmission speed of communication links, and the degree of matching between tasks and virtual machines. A task that runs well on one type of virtual machine does not necessarily work well on another type of virtual machine. On the contrary, it may work poorly. Therefore, it is important to consider matching service capabilities between tasks and virtual machines.

Definition 3 (Service capability): Service capability is the degree to which a service system is capable of providing a service and is often defined as the maximum output rate of the service system.

From the definition of service capability, the service capability value of VM is between 0 and 1. To discuss the concept of service capability matching, this paper divides service capability into service capability requirements and service capability support. Service capability requirement is specific to a task, i.e., how strongly a task requires service capability. Service capability support is specific to a virtual machine, i.e., the extent to which a virtual machine can provide service capability.

To ensure that tasks are executed in the desired processing manner when assigning virtual machines to tasks with different service capability requirements, the service capability supports of the assigned virtual machines should match the service capability requirements of the tasks as much as possible. The value of user satisfaction with the completion of a task by a VM is the most appropriate metric to judge the service capability matching degree between the task and the VM. Whether a user is satisfied with services provided by a VM is obtained by combining the service evaluation indexes. In a cloud datacentre, the performance of virtual machines varies greatly, such as different computation speed, data processing speed, I/O speed, reliability, availability and so on.

In this paper, we consider only three evaluation metrics for service capability support of virtual machines, computation speed, data processing speed, and I/O speed of VMs. An objective determination method based on entropy weights was used to determine the weights of the three evaluation indexes.

The service capability value which is also called *service capability support* of vm_k is calculated as follows.

$$SS(vm_k) = w_1 \times \widetilde{cs_k} + w_2 \times \widetilde{ds_k} + w_3 \times \widetilde{os_k} \tag{3}$$

where $\widetilde{cs_k}$, $\widetilde{ds_k}$ and $\widetilde{os_k}$ are the normalised values of cs_k, ds_k, and os_k, respectively. w_1, w_2, and w_3 are the weights of the obtained *cs*, *ds*, and *os* evaluation indexes, respectively.

w_1, w_2, and w_3 is estimated using the entropy method, which is essentially calculated using the utility value of the information of the index, and the higher its utility value, the greater its importance to the evaluation, so the weight of j^{th} evaluation indexes is

$$w_j = h_j \Big/ \sum_{h=1}^{3} h_k$$

where $j \in \{1, 2, 3\}$, $0 \le w_j \le 1$.

h_j can be calculated as below.

$$h_j = 1 - e_j$$

e_j is the information entropy of the j^{th} evaluation index, for the sample data matrix $X = \{x_{ij}\}_{m\times 3}$ of the above three evaluation indexes for m service transactions have been obtained, and the initial data need to be dimensionless due to the great differences in the magnitude. For specific values in positive increments, such as calculation speed, data processing speed, and I/O processing speed, the user expects them to be as large as possible, and the data is normalised using min-max normalisation, calculated as follows.

$$y_{ij} = \frac{x_{ij} - \min\{x_{ik}\}}{\max\{x_{ik}\} - \min\{x_{ik}\}}$$

Let the normalised matrix after dimensionless processing be $Y = \{y_{ij}\}_{m\times 3}$. Then the information entropy of each index is as follows.

$$e_j = -K \sum_{i=1}^{m} y_{ij} \ln y_{ij}$$

where $j \in \{1, 2, 3\}$, and K is constant. When m samples are in a completely unordered distribution, $y_{ij} = 1/m$ and $e_j = 1$.

$$e_j = -K \sum_{i=1}^{m} \frac{1}{m} \ln \frac{1}{m} = K \sum_{i=1}^{m} \frac{1}{m} \ln m = K \ln m = 1$$

Thus $K = (\ln m)^{-1}$.

2.5 Service capability requirement

A big data workflow may be composed of multiple tasks, each of which has different functions, roles, and importance in the overall parallel task, and must be treated differently in terms of the degree of their demand for service capabilities. According to the structure of DAG, the service capability requirements of tasks in the workflow is related to the out-degree of task nodes, the critical path of workflow, task priority, and task workload.

1 Out-degree weight

The out-degree of the task node has great impact on the subsequent task nodes, and the successful execution of the task can increase concurrent execution of the subsequent tasks. Therefore, the weight $W_{Od}(v_i)$ of the task node v_i on the out-degree is defined as follows.

$$W_{Od}(v_i)=\begin{cases}\dfrac{od(v_i)}{\max\{od(v_1),od(v_2),...,od(v_n)\}}, & od(v_i)\neq 0\\ \dfrac{\min\{od(v_1),od(v_2),...,od(v_n)\}}{\max\{od(v_1),od(v_2),...,od(v_n)\}} & od(v_i)=0\end{cases}$$

where $od(v_i)$ is the out-degree of the task v_i.

2 Critical path weight

A critical path of workflow is a path from the entry node to the exit node, with the maximum sum of computation costs and communication costs.

The weight coefficient in the critical path is defined as follows.

$$W_{Cp}(v_i)=\begin{cases}1, & v_i\in Cp\\ 0.5, & v_i\notin Cp\end{cases}$$

We consider the nodes in critical path are more import that the nodes not on, thus, the weight coefficients of task nodes located on critical paths are set to 1. The weight coefficients of task nodes on non-critical paths can be a decimal between 0 and 1, considering uniform distribution to be prior distribution, it is set to 0.5.

3 Execution order priority weight

There is an execution order dependency in the DAG, and task v_j^i can only be executed after $pred(v_j^i)$ have been executed. For this reason, the task execution order priority $rank(v_i)$ of task v_i is introduced to distinguish the priority execution order of different tasks in the task graph.

The execution order priority $rank(v_i)$ of task v_i is defined as follows.

$$rank(v_i)=\begin{cases}1 & Succ(v_i)=\varnothing\\ \max\limits_{v\in Succ(v_i)}\{rank(v)+1\} & Succ(v_i)\neq\varnothing\end{cases}$$

The execution order priority weight $W_{Rank}(v_i)$ of the task node v_i is defined as follows.

$$W_{Rank}(v_i)=\exp\left(\frac{rank(v_i)-\max_{j=1}^{n}\left(rank(v_j)\right)}{\max_{j=1}^{n}\left(rank(v_j)\right)}\right)$$

4 Task workload weight

The task workload is divided into task computation workload, data processing workload, and I/O processing workload. Three types of workloads have different scales and units, and data normalisation is required for each dimension to eliminate the differences in different scales and units. Suppose a big data workflow contains n tasks, and the task computation workload, data processing workload, and I/O data

processing workload of the three dimensions of the n tasks are represented by a matrix $A_{n\times 3}=(a_{ij})_{n\times 3}$ with n rows and three columns. The z-score standardisation process is performed for each dimension, and the resulting data will conform to a standard normal distribution.

$$a_{ij}^{*}=\frac{a_{ij}-\mu_j}{\sigma_j},\ i\in\{1,2,...,n\},\ j\in\{1,2,3\},$$

$$\mu_j=\frac{1}{n}\sum_{i=1}^{n}a_{ij},$$

$$\sigma_j=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_{ij}-\mu_j)^2}.$$

The workload weight $W_{Workload}(v_i)$ of the task node v_i is defined as follows.

$$W_{Workload}(v_i)=w_1\times a_{i1}^{*}+w_2\times a_{i2}^{*}+w_3\times a_{i3}^{*}$$

where w_1, w_2, and w_3 are the weights of the obtained *cs*, *ds*, and *os* evaluation indexes, respectively.

The service capability requirement $SR(v_i)$ of the task v_i is calculated as follows.

$$SR(v_i)=W_{Od}(v_i)\times W_{Cp}(v_i)\times W_{rank}(v_i)\times W_{Workload}(v_i).$$

3 Cloud-SDLS algorithm

According to the service capability model presented in last section, this paper propose the service dynamic level scheduling algorithm in cloud datacentre (Cloud-SDLS) to extends the traditional DLS algorithm by considering the service capability requirement of tasks and service capability support of virtual machines.

The service dynamic level is defined as follows:

$$SDL(v_i,vm_j)=\left[SS(v_i,vm_j)\right]^{\frac{SR(v_i)}{\max\{SR(v_k)\}}}\left(SL(v_i)-\max\left\{t_{i,j}^{A},t_j^{M}\right\}+t_i^{E}-t_{i,j}^{E}\right)$$

where $SS(v_i, vm_j)$ is the service capability support of vm_j when v_i is scheduled on vm_j. $SR(v_i)$ is service capability requirement of the task v_i for a VM. $\max\{SR(v_k)\}$ is the maximum of service capability requirements of tasks of a big data workflow. $SL(v_i)$ is the static level of the task, $\max\{t_{i,j}^{A}, t_j^{M}\}$ is the time when the task v_i can begin execution on the virtual machine vm_j, $t_{i,j}^{A}$ denotes the time when the data will be available if the task v_i is scheduled on the virtual machine vm_j, and t_j^{M} denotes the time when the virtual machine vm_j will be available for the execution of the task v_i. t_i^{E} denotes the average execution time of the task v_i on all the free virtual machines, and $t_{i,j}^{E}$ denotes the execution time of the task v_i on the virtual machine vm_j.

The proposed Cloud-SDLS algorithm is very scalable and can meet different kinds of service capability support requirements.

Algorithm service dynamic level scheduling algorithm in cloud datacentre (Cloud-SDLS)

Inputs:

$DAG = (V, E, Q, D, O, C)$: a big data workflow

$Cloud = (VM, CS, DS, OS, L, B)$: a cloud datacentre

Outputs:

$Assign = \{(v_i, vm_j)\}$: the set of task's placements

T_{total}: completion time of big data workflow

procedure Cloud-SDLS

SDLS ← $\{SDL(v_i, vm_j), vm_j \in VM, v_i \in DAG, 1 \le i \le n, 1 \le j \le m\}$

$L \leftarrow \{v_i | \text{ indegree}(v_i) = 0, 1 \le i \le n\}$

Assign ← Φ

$\varepsilon \leftarrow L$

$t_s(v_i) = 0, t_e(v_i) = 0, 1 \le i \le n$

$\tau_{idle}(vm_j) = 0, 1 \le j \le m$

$T_{\text{total}} = 0$

do until $\varepsilon = \Phi$

for each $v_i \in \varepsilon$

$(v_i, vm_j) \leftarrow \text{argmax}_{vi,vmj}$ SDLS

$Assign \leftarrow Assign + \{(v_i, vm_j)\}$

$\varepsilon \leftarrow \varepsilon - \{v_i\}$

VM ← VM – $\{vm_j\}$;

$t_s(v_i) = \max\{t_s(v_i), \tau_{idle}(vm_j)\}$;

$T_{Comm}(v_i) = \max\{ct_{k,i} | \ v_k \in \text{pred}(v_i)\}$;

$$t_e(v_i, vm_j) = t_s t_e(v_i, vm_j) + T_{Comm}(v_i) + \frac{q_i}{cs_{vm_j}} + \frac{d_i}{ds_{vm_j}} + \frac{o_i}{os_{vm_j}};$$

$\tau_{idle}(vm_j) = t_e(v_i, vm_j)$;

for each immediate successor v_x of task v_i

$t_s(v_x) = \max(t_s(v_i), \tau_{idle}(vm_j))$

indegree(v_x) = indegree(v_x) – 1

if indegree(v_x) = 0

$\varepsilon \leftarrow \varepsilon + \{ v_x \}$

end if

end for

VM ← VM + $\{vm_j\}$

end for

loop

$T_{total} = \max\{t_e(v_i), 1 \le i \le n\}$

4 Experiment results and analysis

4.1 Experiments environment and configuration

To validate the performance of our algorithm, we build a simulation environment of a cloud computing system with CloudSim 3.0 (Calheiros et al., 2009). CloudSim 3.0 is developed by Java, which supports cloud computing resource management and scheduling simulation. This simulation program runs on a ThinkPad with Intel(R) Core(TM) i7-5500, 2.4GHz, 8 GB, Windows 7 64-bit operating system. In the below experiment, the average communication time between a task and its successor tasks is set to the average execution time of the task multiply CCR (the communication to computation ratio).

Figure 3 Comparison of scheduling length of Cloud-SDLS, DLS, HEFT under different CCR

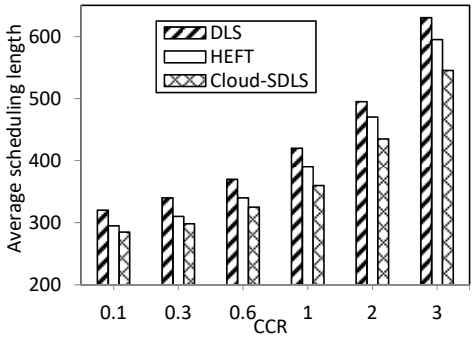
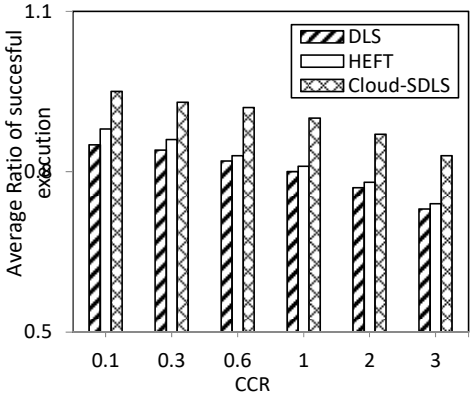


Figure 4 Comparison ratio of successful execution of Cloud-SDLS, DLS, HEFT under different CCR



4.2 Experiment 1: Cloud-SDLS vs. DLS and HEFT

This experiment evaluates the performance of our proposed Cloud-SDLS algorithm. The amount of task nodes and edges in generated workflow graph are both set to 200. We compare traditional DLS and HEFT with Cloud-SDLS in the scheduling length and the

ratio of successful execution, with CCR as 0.1, 0.3, 0.6, 2, 4, or 8. Figures 3 and 4 show the results.

Figure 3 shows that Cloud-SDLS is more scalable for different CCR and performs better than DLS and HEFT methods in different CCR.

In Figure 4, the ratio of successful execution of Cloud-SDLS is much higher than DLS and HEFT. This indicates that the service capability requirement and service capability support mechanism based Cloud-SDLS algorithm makes successful schedule of tasks and the requirement of tasks' service capability requirement to be better satisfied.

4.3 Experiment 2: varying number of tasks

In experiment 2, CCR is set to 1.5, we generate workflow graph with 50 to 120 tasks. We use 200 VMs with 300 links. The comparison result of Cloud-SDLS with DLS and HEFT in the scheduling length and the ratio of successful execution is shown in Figure 5 and Figure 6.

Figure 5 Comparison of scheduling length of Cloud-SDLS with DLS and HEFT under a varying number of tasks

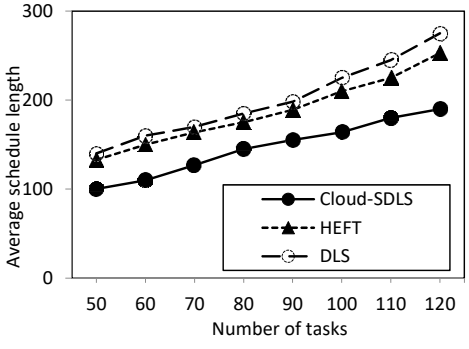
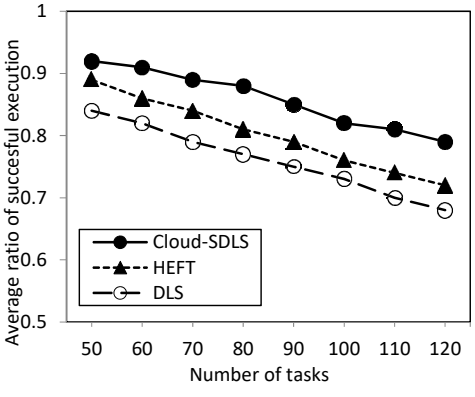


Figure 6 Comparison ratio of successful execution of Cloud-SDLS with DLS and HEFT under a varying number of tasks



As shown in Figure 5, when the number of tasks increases, the average scheduling length of all three algorithms increases, but the scheduling length of Cloud-SDLS is smaller than that of both HEFT and DLS. This is because Cloud-SDLS takes into account both the service capability requirement of the tasks and the service capability support of the VMs. When the dynamic levels of a task on two VMs are equal, the VM with larger service support capacity has a larger service dynamic level. Cloud-SDLS algorithm selects the VM with the largest service dynamic level for a task, which can improve the speed and success rate of the task execution. As a result, the scheduling length of Cloud-SDLS is always smaller than that of HEFT and DLS in scheduling length.

Figure 6 shows that the average ratio of successful execution of Cloud-SDLS is much higher than that of HEFT and DLS, which also verifies that selecting the VM with larger service capacity support to execute a task can improve the execution success rate.

4.4 Experiment 3: varying number of VMs

In experiment 3, CCR is set to 1, the number of VMs from 200 to 500 is created randomly, and the task graph has 300 tasks. The experiment results are shown in Figure 7 and Figure 8. As the number of VMs increases, smaller scheduling length and a higher ratio of successful execution.

Figure 7 Comparison of scheduling length of Cloud-SDLS with DLS and HEFT under a varying number of VMs

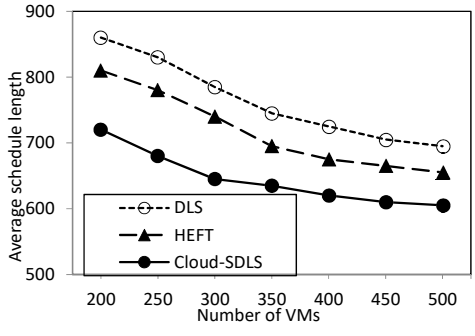
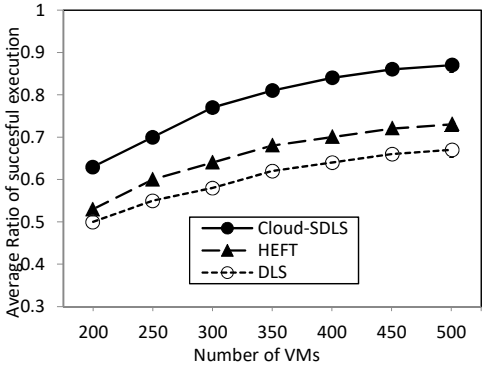


Figure 8 Comparison ratio of successful execution of Cloud-SDLS with DLS and HEFT under a varying number of VMs



5 Conclusions and future work

This paper proposes a metric formula for the service capability requirement of every task in big data workflows and a metric formula for the service capability support of every VM in a cloud datacentre based on the weights of service capability evaluation indexes. The service capability requirement of the task and the service capability support of the virtual machine are merged into the DLS algorithm to obtain the service dynamic level scheduling algorithm in the cloud datacentre (Cloud-SDLS). However, the cloud computing environment is dynamically changing, and how to adaptively adjust the weights of the service capability evaluation indexes according to the changes of the cloud environment is the content to be studied in the future.

Acknowledgements

Sponsored by the National Natural Science Foundation of China (Grant No. RJX20190057), the project of science and technology of the Henan province, China (Grant No. 172102210540, 202102210149), the breakthrough project of social development science and technology of Shanghai, China (Grant No. 21DZ1205000), and the Research Fund for the Doctoral Program of Zhengzhou University of Light Industry, China (Grant No. 2016BSJJ042).

References

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A. and Kennedy, K. (2005) 'Task scheduling strategies for workflow-based applications in grids', in *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp.759–767.

Braun, T.D., Siegel, H.J., Beck, N. et al. (2001) 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', *Journal of Parallel and Distributed Computing*, Vol. 61, No. 6, pp.810–837.

Calheiros, R.N., Ranjan, R., Rose, C.A.F.D. et al. (2009) *Cloudsim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*, ArXiv Preprint, arXiv: 0903.2525.

Chen, H.K., Wen, J.M., Witold, P. and Wu, G.H. (2020) 'Big data processing workflows oriented real-Time scheduling algorithm using task-duplication in geo-distributed clouds', *IEEE Transactions on Big Data*, Vol. 6, No. 1, pp.131–144.

Chen, W. and Xiao, W. (2019) 'Cost-efficient task Scheduling for parallel applications on heterogeneous cloud environment', in *Proceeding of IEEE 21st International Conference on High Performance Computing and Communications*, pp.1651–1657.

Dai, W., Qiu, L., Wu, A. and Qiu, M. (2018) 'Cloud infrastructure resource allocation for big data applications', *IEEE Transactions on Big Data*, Vol. 4, No. 3, pp.313–324.

Gao, Z.W. and Zhang, K. (2012) 'The research on cloud computing resource scheduling method based on time-cost-trust model', in *Proceeding of the 2nd International Conference on Computer Science and Network Technology*, pp.939–942.

Ghafarian, T. and Javadi, B. (2015) 'Cloud-aware data intensive workflow scheduling on volunteer computing systems', Future *Generation Computer Systems*, Vol. 51, No. 1, pp.87–97.

Rjoub, G., Bentahar, J. and Wahab, O.A. (2020) 'Big trust scheduling: trust-aware big data task scheduling approach in cloud computing environments', *Future Generation Computer Systems*, Vol. 110, pp.1079–1097.

Smith, S.P. (2020) 'Cost-efficient dynamic scheduling of big data applications in apache spark on the cloud', *The Journal of Systems and Software*, Vol. 162, No. 1, pp.1–14.

Sun, D.W., Zhang, G.Y., Yang, S.L., Zheng, W.M., Khan, S.U. and Li, K.Q. (2015) 'Re-stream: real-time and energy-efficient resource scheduling in big data stream computing environments', *Information Sciences*, Vol. 319, No. 1, pp.92–112.

Topcuouglu, H., Hariri, S. and Wu, M.Y. (2002) 'Performance-effective and low-complexity task scheduling for heterogeneous computing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, pp.260–274.

Wang, B., Wu, Y.J., Yang, X.L. and Sun, Q.F. (2017) 'Dependency-driven task scheduling scheme of big data processing', *Journal of Software*, Vol. 28, No. 12, pp.3385−3398.

Wang, W., Zeng, G.S., Tang, D. and Yao, J. (2012) 'Cloud-DLS: dynamic trusted scheduling for cloud computing', *Expert Systems with Applications*, Vol. 39, No. 3, pp.2321–2329.

Wei, Z., Qin, Y.S., Bugingo, E. et al. (2018) 'Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds', *Future Generation Computer Systems*, Vol. 82, No, 1, pp.244–255.

Xu, H.Z., Li, R.F. and Zeng, L.N. (2018) 'Parallel task scheduling for resource consumption minimization with reliability, constraint', *Journal of Computer Research and Development*, Vol. 55, No. 11, pp.2569–2583.