Intelligent Cloud Workflow Management and Scheduling Method for Big Data Applications

Yannian Hu
Weifang University
Hui Wang (conglinwh@wfu.edu.cn)
Weifang University
Wenge Ma
Weifang University

Research

Keywords: Big Data, Cloud Workflow, Cloud Service Resource Combination, Scheduling Optimization

Posted Date: December 18th, 2019

DOI: https://doi.org/10.21203/rs.2.19246/v1

Version of Record: A version of this preprint was published on July 21st, 2020. See the published version at https://doi.org/10.1186/s13677-020-00177-8.

Intelligent Cloud Workflow Management and Scheduling Method for Big Data Applications

Yannian Hu[1,a], Hui Wang[2,b*], Wenge Ma[3,c]

[1]Big Data Buro of Weifang, Weifang 261061, Shandong, China

[2]Weifang University, Weifang 261061, Shandong, China

[3]Shandong Provincial Institute of Modern Educational Science, Weifang 261061, Shandong, China

[a] Email: hu_yannian@163.com

*Corresponding author ([b] Email: conglinwh@wfu.edu.cn)

[c] Email: sdjkyb@126.com

ABSTRACT

With the application and comprehensive development of big data, the need for effective research on cloud workflow management and scheduling is becoming more and more urgent. However, there are currently suitable methods for effective analysis. In order to find out how to effectively manage and schedule smart cloud workflows, the article studies big data from different aspects and draws the following conclusions: Compared with the original JStorm system, the average response time is shortened by up to 58.26%, and the average is shortened. 23.18%; CPU resource utilization increased by 17.96%, an average increase of 11.39%; memory utilization increased by 88.7%, an average increase of 71.16%. In optimizing the dynamic combination of web services, the overall performance of MOACO algorithm and CCA algorithm is better than GA algorithm, and the average performance of MOACO algorithm is better than CCA algorithm. The paper also proposes a cloud workflow scheduling strategy based on intelligent algorithms and adjusting the perceived cloud service resource combination strategy to realize two-layer scheduling of cloud workflow tasks. We have studied three representative intelligent algorithms (ACO algorithm, PSO algorithm and GA algorithm) and designed and improved them for scheduling optimization. It can be clearly seen that in the same scenario, in different test cases the optimal values of the different

algorithms vary greatly. However, the optimal solution curve is substantially consistent with the trend of the mean curve.

KEYWORDS

Big Data, Cloud Workflow, Cloud Service Resource Combination, Scheduling Optimization

INTRODUCTION

Cloud workflow technology is an effective way to achieve process integration in a cloud computing environment. Cloud workflow management can improve and optimize business processes, improve business efficiency, achieve better business process control, improve business process flexibility, and improve customer service quality. Cloud workflows make it easy to build, execute, manage, and monitor cloud computing applications, enabling cloud computing applications to be automated and efficient. Due to the dynamic, distributed, heterogeneous and scalable nature of cloud computing, some methods and techniques of traditional workflow cannot effectively deal with related problems in cloud computing environments, and should be based on the characteristics of cloud computing resources and cloud computing applications. The cloud computing workflow architecture, the dynamic process model, resource management model and dynamic scheduling algorithm of cloud computing workflow are studied.

Recently, many research teams at home and abroad have begun to explore opportunities to use molecular data in cloud computing environments. In [1], the authors quantified the transcriptional expression levels of 12,307 RNA sequencing samples from the cancer cell encyclopedia and the cancer genome map. The author used two cloud-based configurations and examined the performance and cost profiles for each configuration. When processing samples, the article collects performance metrics that help us track the duration of each processing step and quantify the computing resources used in different stages of sample processing. In [2], the authors found that cloud infrastructure provides a rich set of management tasks that can manipulate computing, storage, and network resources in the

cloud. The authors adopted a lightweight, non-intrusive approach that is only applicable to interlaced logs that are widely available in cloud infrastructure. CloudSeer first builds an automaton for each administrative task's workflow according to normal execution, and then checks the log messages for a group of automata to check the workflow differences. The differences found during the inspection indicate potential execution issues that may or may not be accompanied by error log messages. In [3], the authors propose an improved particle swarm optimization (IPSO) to schedule applications in cloud resources. The IPSO is used to minimize the total cost of placing tasks on available resources. The article can obtain the total cost value by changing the communication cost between resources, the task-related cost value, and the execution cost of the computing resources. The results show that the improved algorithm is effective compared to the standard particle swarm algorithm. In [4], the author introduces a process model and resource model for energy-aware workflow scheduling in a cloud computing environment. Then, a host workflow-based cloud workflow execution energy calculation method is proposed. Finally, a scheduling algorithm was developed to improve the energy efficiency of cloud workflow execution. Numerical examples and simulation experiments show the feasibility and effectiveness of the proposed method. In [5], the authors propose an alternative architecture that is designed to be suitable for cloud computing and uses a pure software approach to deploy dynamic software infrastructure. The article begins with an overview of the benefits and limitations of cloud-based systems, including those related to the non-real-time nature inherent in virtualization systems. It then introduces the architecture to overcome these limitations, as well as ways to handle security and scalability.

Incremental processing of large-scale data is an increasingly important issue. Because many similar inputs require a lot of repetitive running processing. The standard programming mode is not designed to handle very small updates. So the new system has been specifically addressed for this issue. Unfortunately, this requires a new programming model that breaks compatibility with existing programs while increasing the burden on the programmer. It is now necessary to develop an incremental update mechanism while maintaining compatibility with previous programs. Large-scale data can be automatically incrementally processed by

leveraging previous calculations, algorithms, and programming languages.

Nowadays, the research on massive data processing technology is in full swing, and a lot of meaningful research has been carried out at home and abroad. In [6], the authors found that big data technology is increasingly used in biomedical and health informatics research. Next-generation sequencing technology processes billions of DNA sequence data per day, and the use of electronic health records (EHR) is recording large amounts of patient data. Big data applications offer new opportunities to discover new knowledge and create new ways to improve the quality of care. The application of big data in healthcare is a fast-growing field, and many new discoveries and new methods have been released in the past five years. In [7], the authors found that the operational cost of media stream application providers can be greatly reduced through flexible resource allocation and centralized cloud management. However, to the best of our knowledge, existing migration solutions do not fully consider viewer information such as hardware conditions. In the article, the author considers the deployment optimization problem called ODP by using the local memory of each viewer. Considering the NP difficulty of calculating the optimal solution, the author turns to a computationally evaluable algorithm. In [8], the authors developed an innovative method to extract valuable pixel categories with similar evolution for specific parameters of interest over a long period of time to obtain valuable comprehensive information. The algorithm has been applied to the time series of satellite turbidity products in the coastal areas of the Danube Delta, with a total of approximately 5,700 independent scenarios. Unsupervised classification is performed by the original custom method, which can be executed on such a huge data set. In [9], the authors found that big data applications (such as medical imaging and genetics) typically generate a data set that consists of observations n for more variables p. The author considers the classification problem of $p >> n$ data and proposes a classification method based on linear discriminant analysis (LDA). The proposed method estimates the covariance by using a sparse version of the noise principal component analysis (nPCA). In the experiment, the new method was compared to the latest method for big data problems using simulated data sets and imaging genetic data sets. In [10], the authors found that the increase in the size and complexity of Internet big data has brought unprecedented

opportunities to network physical systems (CPS). The incompleteness and incredibility of Internet big data is challenging to determine the scope of the event. To solve the above problems, the authors proposed the Electrophysical Spatial Event Model (CPSEM) to analyze the impact of events in multiviewers. In addition, the authors also proposed an event impact range detection algorithm (EISDA) to detect the range of hot events in cyberspace and physical space.

In order to find an effective management and scheduling method for intelligent cloud workflow, the article studies big data from different aspects. Based on the existing real-time big data processing open source platform JStorm, the resource dynamic scheduling system D-JStorm is designed and implemented. This paper proposes a cloud workflow scheduling strategy based on intelligent algorithm and adaptive cloud service resource combination strategy, realizes two-layer scheduling of cloud workflow tasks, and studies three representative intelligent algorithms (ACO algorithm, PSO algorithm and GA algorithm). Designed and improved for scheduling optimization, according to the nature of these three algorithms, we propose a scheduling strategy based on intelligent algorithm workflow activities.

METHOD

A. Management combination model based on ant colony algorithm

(1) QoS assessment of management portfolio

Let $WS = \{WS_i \mid i = 1, 2, \ldots, n\}$ be a set of n types of subtasks that the web management combination needs to complete, $ws_j = \{ws_{ij} \mid j = (1, 2, \ldots, m_1)\}$ is a candidate web service class in the UDDI that can complete the subtask WSi, and mi is the number of services in the service class. Let $I_i = \{t_i, c_i, r_i, \ldots\}$ evaluate the set of QoS evaluation indicators of service class wsi, ti is the time index, ci is the price index, ri is the reliability index, and the ellipsis is the scalable quality indicator. Each service class indicator set is different, and ti, ci, and ri can

be used as a web service to dynamically combine the public evaluation indicators of each service class, that is, QoS=execution time, execution cost, and reliability.

Definition 1 Execution time. Let T(wsi) be the execution time of the service subtask, then d $WS_{QoS}^{Tinx}=\sum_{i=1}^{n}T\left(ws_i\right)$ is the execution time of the discovery process. When a subtask is executed sequentially for several service components, then $T\left(w_i\right)=\sum_{j=1}^{k}ws_j$; when the subtask is executed in parallel for several service components, then $T\left(ws_i\right)=\max\left(T\left(ws_j\right)\right)\quad j=1,2,\ldots,k$.

Definition 2 Execution costs. Let C(wsi) be the execution cost of the web service subtask, then $WS_{Q_0S}^{Cost}=\sum_{i=1}^{n}C\left(ws_i\right)$ is the execution cost of the web discovery process.

Definition 3 Service reliability. Let R(wsi) be the service reliability of the service subtask, and $WS_{QaS}^{reliablity}=\prod_{l=1}^{n}R\left(ws_i\right)$ be the reliability of the discovery process.

(2) Multi-objective ant colony algorithm

Since the dynamic combination problem of the web service is to select a suitable service instance from the candidate services for each sub-task discovered by the web, the pheromone of the ksij is selected as τij for the subtask tki, and the heuristic information nij of ksij is selected for the subtask tki. When the algorithm is initialized, the initial value $\tau_{ij}=\tau_0,\quad 1\le i\le n,1\le j\le m$ is set for the pheromone. Multiple QoS parameters with different characteristics are considered in the model. In order to perform multi-objective optimization, different heuristic information needs to be set.

1) Reliability Priority Heuristic Information: RP heuristic information guides ants to select highly reliable web service instances. If the ant's heuristic type is RP, the heuristic information for selecting ksij for the subtask tki can be expressed as:

$$\eta_{ij} = RP_{ij} = \frac{ks^j \cdot r - min\_reliability_i + 1}{max\_reliability_i - min\_reliability_i + 1} \quad (1)$$

Among them, $\min_{reliability_t} = \min_{1 \le 5.5 m_1} \left\{ ks_i^j, r \right\}, \max_{reliability_t} = \max_{1s, sm_1} \left\{ ks_i^j, r \right\}$. This formula ensures that heuristic information is normalized to the (0,1) interval, and the higher the reliability of the web service instance, the greater the value of the heuristic information.

2) Time-prioritized heuristic information (Time Priority): The TP heuristic information guides the ant to select a web service instance with a short execution time. If the ant's heuristic type is TP, the heuristic information for selecting ksij for the subtask tki can be expressed as:

$$\eta_{ij} = TP_{ij} = \frac{max\_time_i - ks_i^j \cdot t + 1}{max\_time_i - min\_time_i + 1} \quad (2)$$

Among them, $min\_time_i = min_{1 \le j \le m_i} \left\{ ks_i^j \cdot t \right\}_i, max\_time_i = max_{1,, j \le m_1} \left\{ ks_i^j \cdot t \right\}$. This formula ensures that heuristic information is normalized to the interval (0, 1), and the higher the reliability of the web service instance, the greater the heuristic information value.

3) Cost Priority Heuristic Information: The CP heuristic information guides the ant to select a web service instance with a short execution time. If the ant's heuristic type is TP, the heuristic information for selecting ksij for the subtask tki can be expressed as:

$$\eta_{ij} = CP_{ij} = \frac{max\_cost_i - ks_i^j \cdot t + 1}{max\_cost_i - min\_cost_i + 1} \quad (3)$$

Among them, $min\_\cos t_i = min_{1 \le j \le m_i} \left\{ ks_i^j c \right\}, max\_cast_i = max_{1 \le j \le m_1} \left\{ ks_i^j \cdot c \right\}$. This formula ensures that heuristic information is normalized to the interval (0, 1), and the higher the reliability of the web service instance, the greater the value of the heuristic information.

B. Resource combination scheduling model for big data processing

(1) Parameter definition

Definition 1 Assume that the limited set of physical clusters in the current streaming big data processing platform is $N=\{N_1,N_2,\ldots,N_d\}$, and the resource configuration of each physical machine is $N_d=<total_d^{cpu},\ total_d^{mem}>$. In order to determine the quantitative indicators in the combined scheduling strategy, it is necessary to quantify the resource utilization of each node. This topic uses different computing resources of the CPU and memory to perform scheduling, and quantifies the resource utilization rate on each node.

Definition 2 Node resource utilization Ud the ratio of the actual resource amount occupied by the resource utilization on each node to the total resource amount of the node when running on the node, the CPU and memory resource utilization on the node are as follows The formula is derived.

$$U_d^{cpu}=\frac{\sum_{j=1}^{n}R_{dj}^{cpu}}{total_d^{cpu}}$$
$$U_d^{mem}=\frac{\sum_{j=1}^{n}R_{dj}^{mem}}{total_d^{mem}} \quad (4)$$

Among them, Ucpud and Umemd respectively represent the CPU and memory resource utilization of the physical node Nd, and $\sum_{j=1}^{n}R_{dj}^{cpu}$ and $\sum_{j=1}^{n}R_{dj}^{mem}$ respectively represent the sum of the memory and CPU resource usage of the computing container running on the physical node Nd.

(2) Scheduling operation timing

For a given calculation container, you can first determine whether the calculation container requires resource rescheduling. The judgment rule is as shown in the formula:

$$PR_{i(n+1)}^{cpu}\neq AR_{in}^{cpu}$$
$$PR_{i(n+1)}^{mem}\neq AR_{in}^{mem} \quad (5)$$

Where $PR_{i(n+1)}^{cpu}$ and $PR_{i(n+1)}^{mem}$ represent the CPU and memory resource prediction values for the n+1th time window of the i-th calculation container, respectively, and ARcpuin and ARmemin represent the CPU and memory resources of the i-th calculation container in the nth time window, respectively. The actual assigned value. That is, as long as the actual allocated resource amount is different from the predicted amount, resource rescheduling is performed on the computing container, and the computing container is added to the resource rescheduling queue RSQ.

(3) Calculation of resource increase and decrease

First, the resource prediction value $PR_{i(n+1)} = \left(PR_{i(n+1)}^{cpu}, PR_{i(n+1)}^{mem} \right)$ of the i-th calculation container and the resource configuration amount $AR_{in} = \left(AR_{in}^{cpu}, AR_{in}^{mem} \right)$ of the i-th calculation container in the n+1th time window are obtained, and the resource adjustment amount $\Delta R_{i(n+1)}$ of the container i in the n+1th time window is calculated.

$$\begin{aligned} &\Delta TR_{i(n+1)} =< \Delta R_{i(n+1)}^{cpu}, \Delta R_{i(n+1)}^{mem} > \\ &\Delta TR_{i(n+1)}^{(pu)} = PR_{i(n+1)}^{cpw} - AR_{in}^{cpu} \\ &\Delta TR_{i(n+1)}^{mem} = PR_{i(n+1)}^{mem} - AR_{in}^{mem} \end{aligned} \quad (6)$$

Note that the resource prediction value of the CPU and memory of each calculation container may be smaller than the current resource configuration amount, or may be greater than the current resource configuration amount. Therefore, when $\Delta TR_{i(n+1)}^{cpu}$ or $\Delta TR_{i(n+1)}^{mem}$ is greater than 0, it is indicated as resource addition to the CPU or memory. When $\Delta TR_{i(n+1)}^{cpu}$, $\Delta TR_{i(n+1)}^{mem}$ is less than 0, it means that the CPU or memory resources are reduced.

C. Cloud workflow related theory

(1) Scene model

The core business process analysis of the platform is as follows:

First, the service requester logs in to the system through a legal user name and password, and starts the service application according to the workflow rules of the company. The application process mainly includes entering the application data, submitting the application, and waiting for the application to be accepted;

Secondly, the acceptor of the acceptance center accepts the service application data information, checks the business data, accepts the service application, fills in the acceptance opinions, and views the workflow;

Once again, the dispatching center dispatcher reviews the business form data information, reviews the acceptor's acceptance result, and distributes the event;

Finally, the dispatching center dispatcher feeds back the audit opinion to the acceptor. The dispatcher distributes the event to the squad leader according to the business demand. The squad leader calculates the formation, waits for the decision maker to issue the order, and the disposal department begins the business implementation process after receiving the instruction. After that, the result of the treatment is fed back to the acceptor, the acceptor summarizes the information, and the result of the processing is fed back to the service requester, and the service requester performs the next event flow, and at the same time, generates a workflow information table, performs the warehousing process, and ends. A workflow message, in which the information maintainer and workflow supervisor maintain and monitor the workflow information at any time.

(2) Role model

The workflow performer role can be abstracted according to its functions in event processing: application requester, service requester, acceptor, dispatcher, squad leader, decision maker and disposal department. Acceptor use cases include: information collection, task distribution , acceptance confirmation, programming, emergency watch, incident report and comprehensive coordination, service requester use cases include: service application, information retrieval and alarm, disposal department use cases include: information feedback,

information retrieval and command reception, squad leader use cases include: Information collection, information reporting, task distribution, log management, command reception, and event monitoring. Decision maker use cases include: situation monitoring, program validation, and event monitoring. Scheduler use cases include: task signing, duty management, situation monitoring, and service auditing. Workflow monitor use cases include: situation monitoring, message monitoring, business manager use cases include: business management, user management and personal information management, information maintainer use cases include: information maintenance, backup maintenance and transceiver management, application requester use cases include: Select workflow template, configuration worker flow templates, application configuration and data configuration.

D. Dynamic resource prediction model for big data processing

(1) Parameter definition

This paper introduces a sliding window. For each application, the resource usage prediction value of the i-th calculation container in the n+1-th time window Wn+1 can be expressed as:

$$PR_{i(n+1)} = g\left(R_i\right) \quad (7)$$

Where g(Ri) represents a resource usage prediction model, and for all the computing containers $CC = \{CC_1, CC_2, \cdots, CC_m\}$ of the streaming big data processing platform, the resource usage history data of each computing container CCi is obtained by monitoring in each time window. Set, form a data stream Ri with temporal properties, as defined below:

Definition 1 Physical resource usage sequence

For the i-th calculation container CCi ($i \leq m$), the corresponding resource usage amount in the n-th window is Rin, and the resource usage sequence $R_i = \{R_{i1}, R_{i2}, \cdots, R_{in}\}$ of the calculation container CCi is obtained in the entire time series. Where n is the number of time

windows and Rin is the amount of resources used by the application's i-th calculation container in the nth time window. Since resource usage includes CPU resource usage and memory resource usage, Rin can be expressed as $R_{in} = \left\{ R_{in}^{cpu}, R_{in}^{cpu} \right\}$.

Definition 2 Resource usage change rate sequence

For the i-th calculation container CCi, the difference between the adjacent nth time window and the n-1th time window is expressed as the resource usage change rate $\Delta R_{in} = R_{in} - R_{i(n-1)}$, thereby obtaining the calculation container resource usage change rate sequence $\Delta R_i = \left\{ \Delta R_{i1}, \Delta R_{i2}, \ldots, \Delta R_{in} \right\}$. Since Ri contains both CPU and memory resources, the resource usage change rate sequence includes the CPU change rate sequence ΔR_{in}^{cpu} and the memory change rate sequence $\Delta R_{in}^{mem}, \Delta R_{in} = \left\{ \Delta R_{in}^{cpu}, \Delta R_{in}^{mem} \right\}$, and $\Delta R_{in}^{cpu}, \Delta R_{in}^{mem}$ are calculated by the formula:

$$\begin{aligned} \Delta R_{in}^{cpu} &= R_{in}^{cpu} - R_{i(n-1)}^{cpu} \\ \Delta R_{in}^{\text{mem}} &= R_{in}^{\text{mem}} - R_{i(n-1)}^{\text{mem}} \end{aligned} \quad (8)$$

(2) Resource prediction model based on resource usage change rate

The resource usage prediction value of the i-th calculation container at the n+1th time is calculated by the CPU and memory usage history sequence respectively, and can be expressed by the formula as:

$$g\left(R_i\right) = f\left(R_i^{cpu}, R_i^{mem}\right) \quad (9)$$

Where, as shown in the definition 1, R_i^{cpu} is the sequence of CPU resource usage from the start of the i-th calculation container to the nth time window, and R_i^{mem} is the sequence of corresponding memory resource usage. The resource usage sequence is volatility and continuity, so it can be based on the CPU and memory resource usage of the nth time window

of the i-th calculation container to accumulate or reduce the usage change rate. The resource prediction value of the n+1th time window, the problem is converted into a formula:

$$f\left(R_i^{cpu}, R_i^{mem}\right) = \left\{R_i^{cpu} + \Delta R_{i(n+1)}^{cpu'}, R_i^{mem} + \Delta R_{i(n+1)}^{mem'}\right\} \quad (10)$$

Since the CPU resource usage R_{in}^{cpu} and memory resource R_{in}^{mem} usage of the nth time window are known, the problem translates into how to find the resource usage change rates $\Delta R_{i(n+1)}^{cpu'}$ and $\Delta R_{i(n+1)}^{mem'}$ at the next moment.

EXPERIMENT

A. Test environment

The test environment of the system implemented in this paper consists of 6 physical nodes configured as shown in Table 1, one of which is the master node, four of which are the compute nodes, and one of which serves as the client to simulate the changing user's request and will request the data. The change law of the poisson probability density is continuously sent to the Kafka application over time.

Tab.1 Test environment configuration

Resource Type	Resource Name	Resource Configuration
Hardware	**CPU**	**Intel(R) Core(TM) i5-3470 CPU @ 3.20GHZ * 4**
Software	**RAM**	**8GB**
	External storage	**1TB**
	The internet	**Gigabit Ethernet**
	Operating system	**Centos 6.5**
	JVM	**jdk 1.7.0_65**
	Middleware	**Zookeeper3.4.5**
	Compiler Environment	**Maven 3.2.2**

B. Workflow selection

In the experiment of this paper, five workflow aggregates, which are representative in the field of workflow research, are selected, namely SIPHT, LIGO, Epigenomics, Montage and CyberShake. Among the five workflow assemblies, each workflow has a different structure, different data sources and computational characteristics.

Among them, SIPHT is derived from the Harvard Bioinformatics Project. It is an automated search for the sRNA-encoding genes of all bacteria in the database of the International Bioinformatics Center. The tasks in the workflow require higher CPU processing power and I/O requirements. LIGO workflow comes from the field of physics, the goal is to analyze and detect gravitational wave data, is a CPU-intensive task, and consumes a lot of memory. The Epigenomics workflow is used to automate various operations in genome sequence processing and requires strong CPU processing power. The Montage workflow comes from NASA/IPAC and is an astronomical application for generating specific mosaics based on input images. Most of its workflow is I/O intensive and does not require a lot of CPU processing power. CyberShake is a data-intensive workflow that is generated by the South Califonia Earthquake Center to generate seismic hazards. It also requires a lot of memory and CPU support. These five workflows are used to represent five different applications in the experiments in this paper. In each application, the workflow consists of 50/100/200/300/400/500/600/700/800/900 and 1000 tasks, respectively. The parameters and runtime time of each workflow are determined based on the real workflow log.

C. Performance evaluation indicators

(1) Forecast performance evaluation index

This paper uses Absolute Error (AE) to measure. AE is used to measure the difference between the CPU or memory resource usage forecast and the actual usage within a single time window, as calculated by the following formula.

$$AE = X_{observe,n} - X_{predict,n} \quad (11)$$

In the formula, Xoberve,n represents the observation of the CPU or memory resource usage of the compute container in the nth time window, and Xpredict,n represents the predicted CPU or memory resource usage of the compute container in the nth time window.

(2) Overall performance evaluation index

The time-sensitive demand satisfaction rate is expressed as the ratio of the number of successfully processed data within the deadline and the total number of requests sent during the period over a period of time. The calculation method is as shown in the formula:

$$P_{qos} = \frac{K(T)}{N(T)} \quad (12)$$

Where N(T) represents the total amount of data arriving within the T time window, and K(T) represents the number of data successfully processed within the deadline within the T time window.

The application processing response time is expressed as the average value of the time spent successfully processing each data in the unit time. The total number of processing data per unit time can be expressed by the number of tuples per unit time, each data processing time. The difference from the time the data is sent from the application to the time it is fully processed by the application and acked.

$$L\left(T_k\right) = \frac{\sum_{i=0}^{n\left(T_k\right)} L_i}{n\left(T_k\right)} \quad (13)$$

Where Tk represents the current Kth time window, n(Tk) represents the sum of the processed data amount in the Tk time window, and Li represents the processing delay time of the i th data in the Tk time window.

The application resource utilization is expressed as the ratio of the amount of CPU or memory resource Rallocate allocated to the application to the actual resource usage Ruse in the unit time window. In the entire steady-state time period T of the application, the application within the time period is used. The sum of resource usage is then averaged to represent the average resource utilization of the application. The calculation formula is as shown in the formula:

$$R(T)=\frac{1}{n}\sum_{w=1}^{n}\frac{R_{use,w}}{R_{albcate}} \quad (14)$$

What is calculated by this formula is the value of resource usage within a time window w.

RESULTS AND DISCUSSIONS

A. Performance analysis under different data arrival strengths

In this paper, the SequenceTest program is used for testing. The data arrival rate is divided into three groups according to the order from low to high. As shown in Table 1, the horizontal header indicates the parameter value test group number, and the column indicates the data arrival rate under the group number, tps/s represents the average number of Tuples sent per second. The initial resource allocation defaults to <1Core, 2GB> for each computing container. This value is the default resource configuration for the JStorm cluster. The time guarantee requirement is 400ms, which is the average data arrival rate in the default configuration.

Tab.2 Data arrival strength change grouping

Value group	Bin 1	Bin 2	Bin 3
Average arrival rate	420 tps/s	1120 tps/s	2160 tps/s

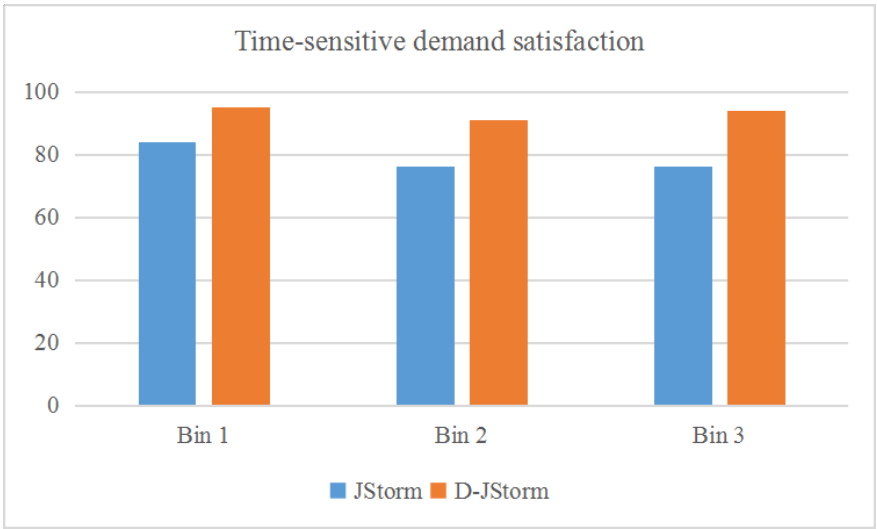
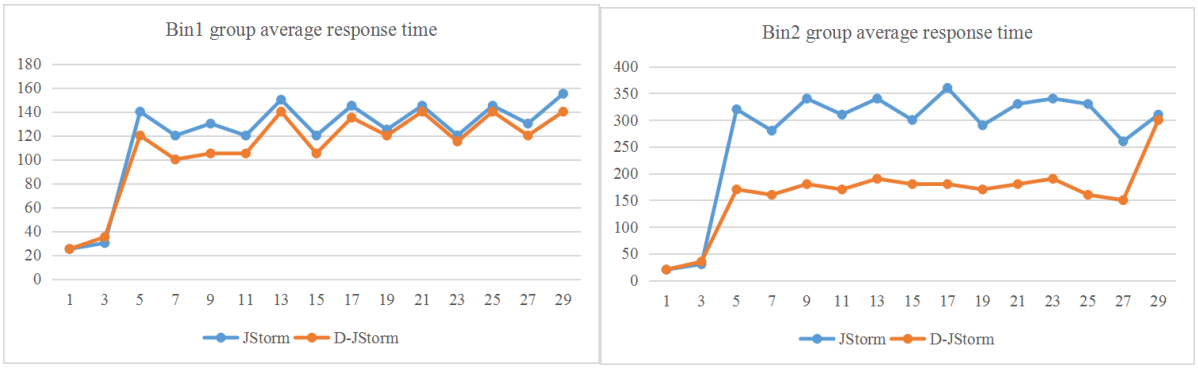


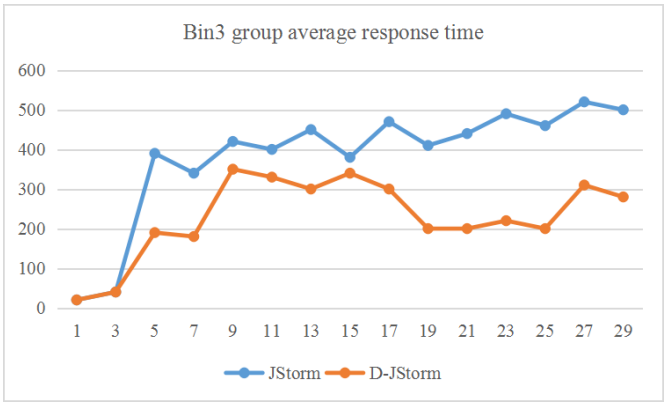
Fig.1 Comparison of the satisfaction of time-dependent demand under different data arrival strengths

In terms of timeliness requirements, as shown in Figure 1, compared with the JStorm system, the maximum increase was 15.26%, the lowest increase was 11.35%, and the average increase was 13.38%.

Assuring timeliness is the primary goal of a streaming big data processing platform, and response time is a direct response to time-sensitive requirements. The improvement of timeliness requirements in this paper is mainly reflected in the fact that when the application load reaches the peak, it can dynamically allocate enough CPU and memory resources for each computing container in advance, so that the computing container can have enough calculation and storage at the peak of the demand resources. At the same time, at the time of the load downturn, due to the reduced demand for resources, the arriving data requests can be processed in sufficient time. Based on the above two factors, the timeliness of data processing due to the resource pre-feeding at the peak time and the resource demand at the load trough time are maximized.



(a) Average response time curve of Bin1 group (b) Average response time curve of Bin2 group



(c) Average response time curve of the Bin3 group

Fig.2 Comparison of average response time under different data arrival intensities

In terms of response time, as shown in Figure 2, compared with the JStorm system, the minimum is shortened by 15.69%, the maximum is shortened by 35.1%, and the average is shortened by 26.76%.

The performance improvement of the response time compared with the original JStorm system is mainly reflected in the peak load time. This paper selects each node to run a calculation container to experiment, avoiding the interaction between multiple computing containers, due to computing resources and storage resources. Dynamic allocation, so that a single computing container can get more resources to calculate at the peak load time, the data dwell time in the cache queue is reduced, so that the amount of data queued in the queue is reduced, and each data is in each component. The processing time is also greatly reduced, which in turn leads to a reduction in the overall data from the time it flows into the system until it is completely processed.

B. Performance analysis under different initial resource configurations

In order to analyze the impact of the initial resource allocation on the D-JStorm system, this paper divides the initial resource allocation of different computing containers into three groups, which are divided into three categories: resource shortage, default resource configuration and resource affluence, as shown in Table 3.

Tab. 3 Test load configuration grouping for each compute unit

Value group	Bin 1	Bin 2	Bin 3
Core	0.5	1	4
CPU	2	2	8

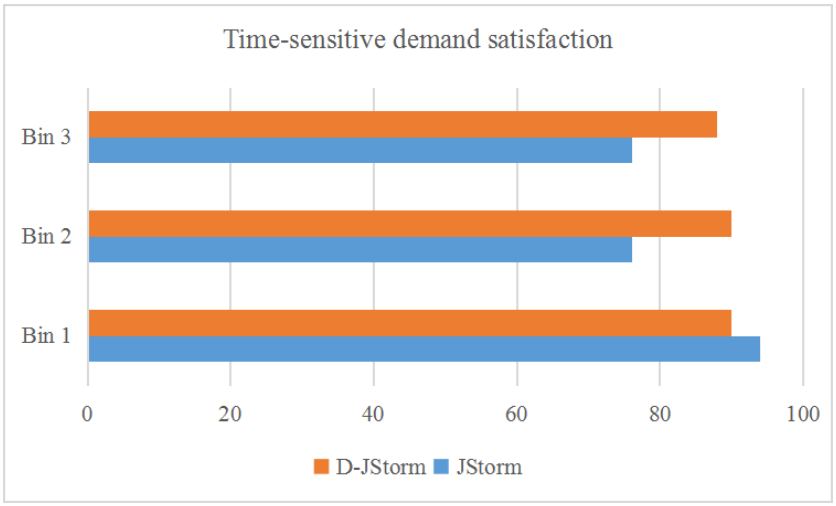
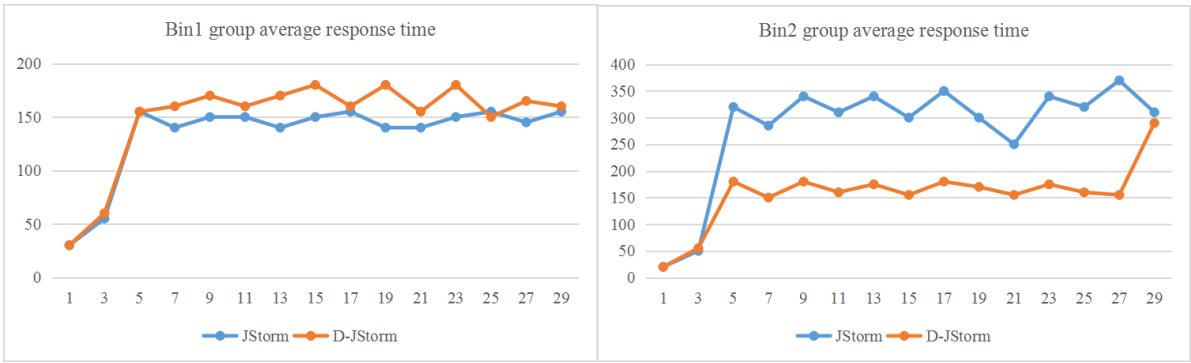


Fig.3 Comparison of time-based demand satisfaction degree under different initial resource configurations

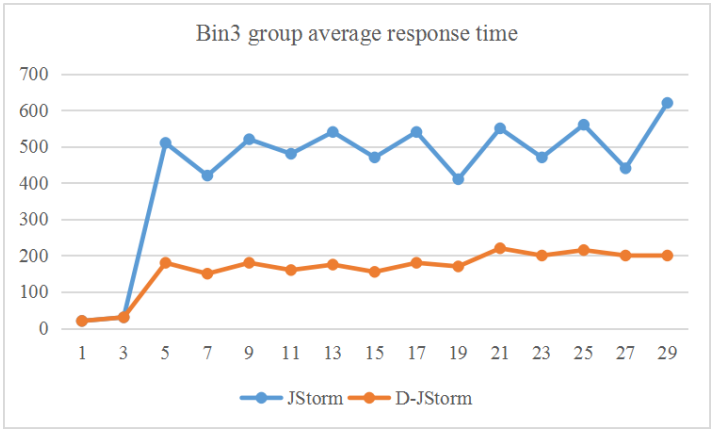
In terms of timeliness requirements, as shown in Figure 3, compared with the JStorm system, except for the Bin1 group, which decreased by 4.17%, the Bin2 group increased by 13.5%, and the Bin3 group increased by 11.16%, with an average increase of 6.83%.

For the Bin1 group, the initial resource configuration <CPU, memory> corresponds to <0.5Core, 2GB>, and the CPU resources are resource-intensive resources. Since the prediction method selected in this paper is based on the prediction of the historical time window sequence, it needs to have certain guaranteed time-sensitive resource allocation as a

historical experience value, while the configuration of 0.5 cores keeps resources in a state of tension, making the predictive component not have enough low-latency response history values to determine how much should be allocated for a given time-sensitive requirement. Resources cause the system to not get enough resources to process the data at the moment of load increase, which leads to excessive data accumulation, increased latency, and reduced timeliness guarantee. Compared with the Bin2 group and the Bin3 group, the CPU resources are given a sufficient amount of resource allocation as compared with the computing resources, so that sufficient historical experience values can be obtained at the start of the program to guide the resource prediction. At the peak load time, it can ensure that there are enough resources to process the data, reduce the amount of data waiting, and thus improve the timeliness guarantee.



(a) Average response time curve of Bin1 group (b) Average response time curve of Bin2 group



(c) Average response time curve of the Bin3 group

Fig.4 Average response time curve in different initial configurations

In terms of response time, as shown in Figure 4, compared with the JStorm system, except for the Bin1 group, the response time has been improved by 58.26%, and the average is shortened by 27.68%.

The increase in response time directly leads to a reduction in time guarantee. The decrease in response time compared to the JStorm system occurs when the resource configuration is <0.5Core, 2GB>. According to the data analysis, the memory resource is not the bottleneck of the system. Therefore, at the peak load, due to the failure of timely resource supply, the tight allocation of CPU resources as computing resources causes a large amount of data to accumulate in the cache queue, and is also calculated. The processing rate of data in the container is also greatly reduced, which in turn leads to an increase in the response time of the entire data set. As can be seen from the Bin2 group and the Bin3 group, as the initial resource allocation increases, the average response time of the system also increases gradually. This is because the sufficient initial resource configuration enables the prediction algorithm to have accurate resource requirements and response time. The data set of the relationship is used as the basis for prediction, so that the prediction accuracy is gradually improved, and more resources can be adjusted in time to load more computing containers, so that the processing rate of the data is increased, thereby reducing the response time.

C. Management combination analysis based on ant colony algorithm

Figure 5 shows that the population size is 1000, and the number of web service candidates per service class changes from 1 to 50. The MOACO algorithm's optimal fitness and the actual optimal fitness kiss rate (the ratio of the same fitness value) are 97%. The optimal fitness curve of the algorithm basically coincides with the actual optimal fitness curve. The average number of termination generations is 80.5, the best fitness is 0.868. The optimal fitness and actual optimal fitness of the CCA algorithm are 87%, the average number of termination generations is 90.56, and the optimal fitness is 0.8006. The best fit and actual optimal fitness of the GA algorithm is 79%, the average number of termination generations is 110.43, and the optimal fitness is 0.647. The experimental results show that with the increase

of the number of web service class candidate services, the maximum fitness is generally increasing, which indicates the dynamic combination of web services. As the number of web services published by service provider increases, the WS core can select the optimal services.

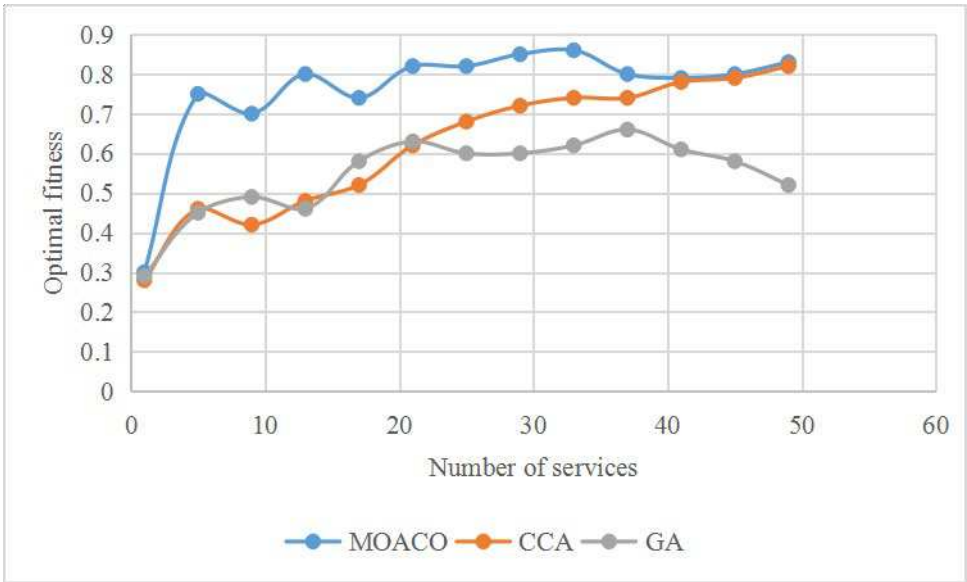


Fig.5 Optimal fitness at different service scales

Adjusting the awareness policy can improve the execution success rate of the cloud workflow. Due to the dynamic nature of the cloud computing environment, web services that execute cloud workflow activities may fail to run, which affects the execution success rate of the entire cloud workflow. In the experiment, six cloud workflow instances with 50, 100, 150, 200, 250, and 300 activities were set up, and each group of cloud workflows was run 50 times, and the average execution success rate was taken. From the experimental results in Figure 6, we can see that the adjustment awareness strategy can significantly improve the execution success rate of the cloud workflow. When the number of activities is 50, the execution success rate of the no-adjustment strategy is 92%, and the execution success rate of the adjustment-aware strategy is 98%. When the number of activities is 150, the execution success rate of the no-adjustment-aware strategy is 89%. The execution success rate of the adjustment-aware strategy is 95%; when the number of activities is 300, the execution success rate of the no-adjustment-aware strategy is 82%, and the execution success rate of the adjustment-aware strategy is 88%. As the number of activities increases, the execution success rate of both strategies decreases. This is because the more activities, the more web

services are needed, and the greater the interaction between services, resulting in successful execution.

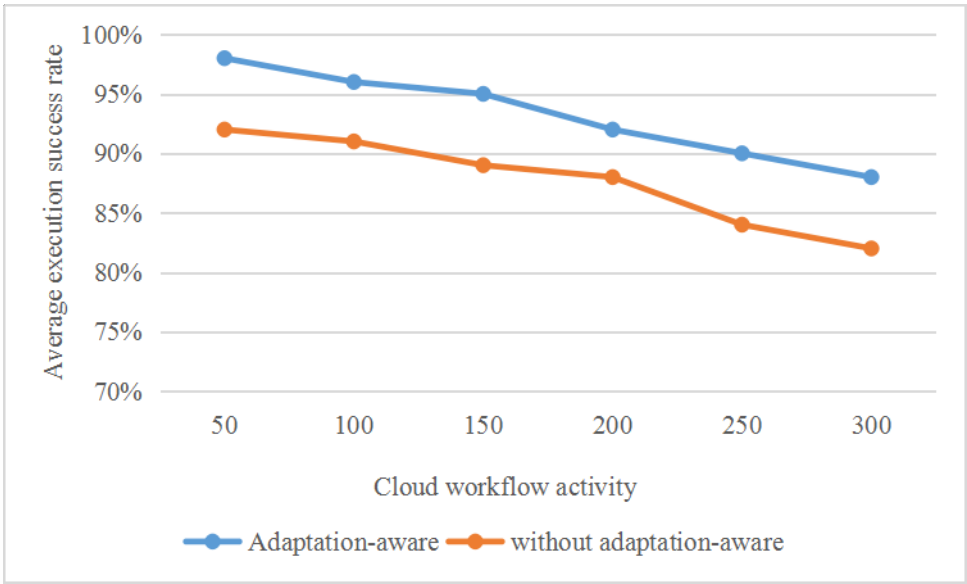


Fig.6 Effect of adjusting the perception policy on execution success rate

D. Cloud workflow activity scheduling analysis

Figure 7 shows the optimization rate for overall completion time in 10 different experimental setups. As can be seen from Figure 7, when the workflow size is small, the completion time optimization rates of GA, ACO, and PSO are similar, but they are all low. For example, in the first set of experiments, the cloud workflow had 50 tasks, and the completion time optimization rate of ACO was about 5%, PSO was about 5.6%, and GA was about 5.8%. When the scale of the workflow increases, the performance in the GA maintains a relatively stable state, but it shows a downward trend; the performance of the PSO is an upward trend at the beginning, and then falls sharply; while the task scale increases, the ACO has the ability. For example, in the last set of experiments, the workflow had 300 tasks, the optimization rate of ACO algorithm completion time was 24.4%, GA was 11.8%, and PSO was only 3.88%. It is worth noting that PSO achieved better performance than the other two algorithms when the workflow scale changed from 150 to 200. For example, when the number of tasks is 180, the optimization rate of the completion time of the PSO is 22.2%, while the GA is 13.8% and the ACO is 15.4%.

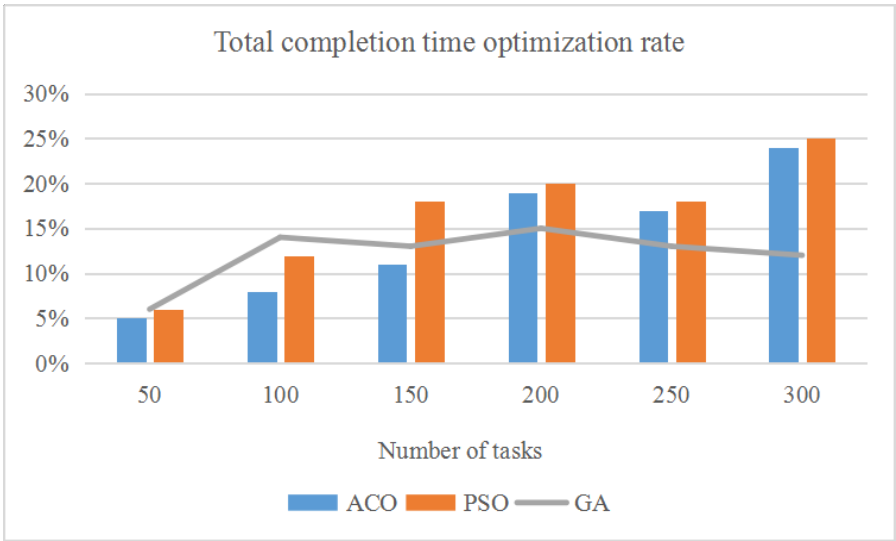


Fig.7 Optimization rate of overall completion time

A unique phenomenon observed is that the ACO algorithm performs better when the number of tasks is greater than 200. This shows that the ACO algorithm is more effective in solving large-scale discrete multi-constrained optimization problems. Probably because the ACO algorithm is a task and then a task to construct an efficient solution, and the PSO algorithm and the GA algorithm randomly search for solutions in the problem solving space. Therefore, the solutions generated by the ACO algorithm construction can satisfy all the constraints, and the GA algorithm and the PSO algorithm cannot guarantee that the generated solutions are feasible solutions. Because of this, the ACO algorithm still has higher performance when the workflow size increases. The GA algorithm and the PSO algorithm exhibit premature convergence due to the limited representation space.

CONCLUSIONS

In order to find ways to effectively manage and schedule intelligent cloud workflows, the article has studied big data from different aspects and reached the following conclusions:

(1) Based on the research results of this paper, based on the existing real-time big data processing open source platform JStorm, the resource dynamic scheduling system D-JStorm is designed and implemented. The performance analysis of the D-JStorm system shows that compared with the original JStorm system, the average response time is reduced by up to

58.26%, and the average is shortened by 23.18%. The CPU resource utilization rate is up by 17.96%, with an average increase of 11.39%. Increased by 88.7%, an average increase of 71.16%.

(2) The average termination algebra of GA is 116.46, the average termination algebra of CCA is 103.9, and the average termination algebra of MOACO is 89.62. It can be seen that the convergence of the GA algorithm is the worst, while the convergence of the MOACO algorithm is the best, and the convergence of the CCA algorithm is between the two. When the number of web services exceeds 43, the growth rate of the termination algebra of the GA algorithm and the CCA algorithm is significantly accelerated, and the growth rate of the GA algorithm is the largest. The MOACO algorithm is slow to grow. This shows that the MOACO algorithm is more suitable for optimizing the large-scale dynamic web service discovery process. Overall, the overall performance of the MOACO algorithm and the CCA algorithm is better than the GA algorithm in optimizing the dynamic combination of web services, and the average performance of the MOACO algorithm is better than the CCA algorithm.

(3) A cloud workflow scheduling strategy based on intelligent algorithm and adaptive cloud service resource combination strategy is proposed to realize two-layer scheduling of cloud workflow tasks. We have studied three representative intelligent algorithms (ACO algorithm, PSO algorithm and GA algorithm) and designed and improved them for scheduling optimization. According to the nature of these three algorithms, we propose workflow activities based on intelligent algorithms. It can be clearly seen that although in the same scenario, the optimal values of the different algorithms vary greatly in different test cases. However, the optimal solution curve is substantially consistent with the trend of the mean curve.

Declarations

Availability of data and materials

All the data and materials in this article are real and available

Competing interests

These no potential competing interests in our paper. And all authors have seen the manuscript and approved to submit to your journal. We confirm that the content of the manuscript has not been published or submitted for publication elsewhere.

Author's contributions

All authors take part in the discussion of the work described in this paper.

Acknowledgements

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions. I would like to acknowledge all our team members.

Author details

Yannian Hu was born in Gaomi，Shandong P.R. China, in 1976. He received the bachelor's degree fromQigndao University. Now, he works in Big Data Buro of Weifang. His research interests include computational intelligence, information security and big data analysis etc..

Hui Wang was born in Gaomi, Shandong P.R. China, in 1978. She received a master's degree from Guangxi Normal University. Now, she works in Weifang University .Her research interests include big data analysis, ideological and political educationand and communication theory etc..

Wenge Ma was born in Yuncheng County, Shandong Province, China in 1987. He received a master's degree from Beijing Technology and Business University. Now he works in Shandong Modern Education Science Research Institute Science .Research interests are AI education and big data technology.

REFERENCES

[1] Tatlow, P. J., & Piccolo, S. R. (2016) "A Cloud-Based Workflow to Quantify Transcript-Expression Levels in Public Cancer Compendia". Sci Rep, 6(1), pp. 39259.

[2] Yu, X., Joshi, P., Xu, J., Jin, G., Zhang, H., & Jiang, G. (2016) "Cloudseer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs". Acm Sigarch Computer Architecture News, 44(2), pp. 489-502.

[3] Sadhasivam, N., & Thangaraj, P. (2016) "Design of An Improved PSO Algorithm for Workflow Scheduling in Cloud Computing Environment". Intelligent Automation & Soft Computing, 23(3), pp. 1-8.

[4] Xie, Y., Tianta, H. E., Qianyun, N. I., & Hanqing, W. U. (2017) “Scheduling for Improving the Energy Efficiency of Cloud Workflow Execution”. Systems Engineering-Theory & Practice, 37(4), pp. 1056-1071.

[5] Cartwright, R. (2018) “An Internet of Things Architecture for Cloud-Fit Professional Media Workflow”. Smpte Motion Imaging Journal, 127(5), pp. 14-25.

[6] Luo, J., Wu, M., Gopukumar, D., & Zhao, Y. (2016) “Big Data Application in Biomedical Research and Health Care: A Literature Review”. Biomedical Informatics Insights, 8(8), pp. 1-10.

[7] Wu, T., Dou, W., Fan, W., Tang, S., Hu, C., & Chen, J. (2016) “A Deployment Optimization Scheme over Multimedia Big Data for Large-Scale Media Streaming Application”. Acm Transactions on Multimedia Computing Communications & Applications, 12(5s), pp. 1-23.

[8] Cucudumitrescu, C., & Constantin, S. (2017) “Extraction of Regions with Similar Temporal Evolution Using Earth Observation Big Data. Application to Water Turbidity Dynamics”. Remote Sensing Letters, 8(7), pp. 627-636.

[9] Ulfarsson, M. O., Palsson, F., Sigurdsson, J., & Sveinsson, J. R. (2016) “Classification of Big Data with Application to Imaging Genetics”. Proceedings of the IEEE, 104(11), pp. 2137-2154.

[10] Xue, Y., Xu, L., Jie, Y., & Zhang, G. (2017) “A Hot Event Influence Scope Assessment Method in Cyber-Physical Space for Big Data Application”. Intelligent Automation & Soft Computing, 24(1), pp. 1-9.

Yannian Hu was born in Gaomi，Shandong P.R. China, in 1976. He received the bachelor‘s degree fromQigndao University. Now, he works in Big Data Buro of Weifang. His research interests include computational intelligence, information security and big data analysis etc..

E-mail: hu_yannian@163.com

Hui Wang was born in Gaomi, Shandong P.R. China, in 1978. She received a master's degree from Guangxi Normal University. Now, she works in Weifang University .Her research interests include big data analysis, ideological and political educationand and communication theory etc..

E-mail: conglinwh@wfu.edu.cn

Wenge Ma was born in Yuncheng County, Shandong Province, China in 1987. He received a master's degree from Beijing Technology and Business University. Now he works in Shandong Modern Education Science Research Institute Science .Research interests are AI education and big data technology.

E-mail: sdjkyb@126.com

Figure

Fig.6 Effect of adjusting the perception policy on execution success rate

Fig.7 Optimization rate of overall completion time

Figures

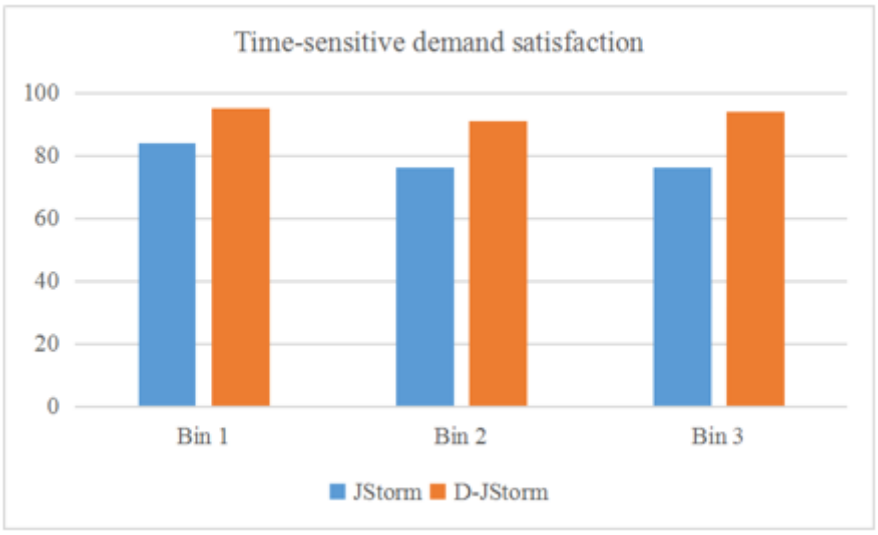
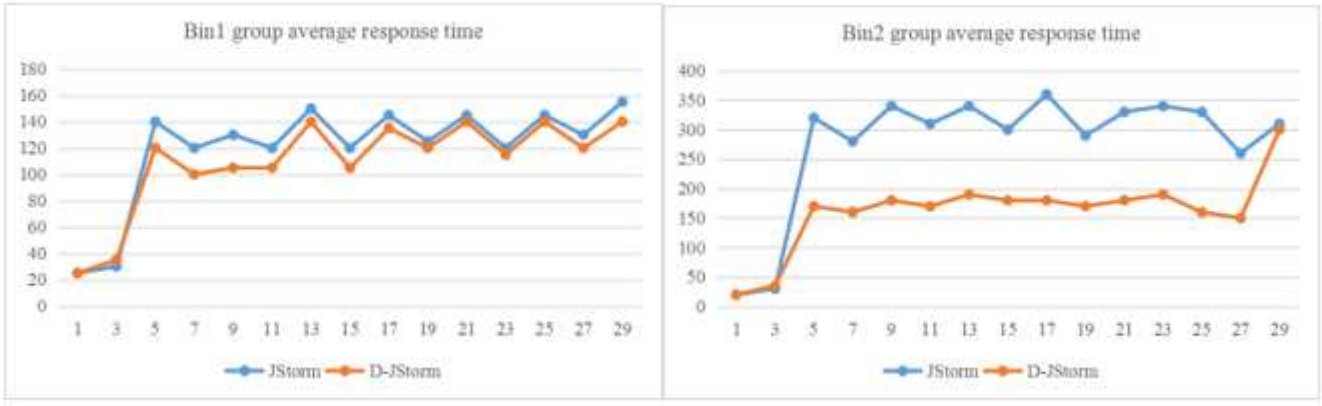
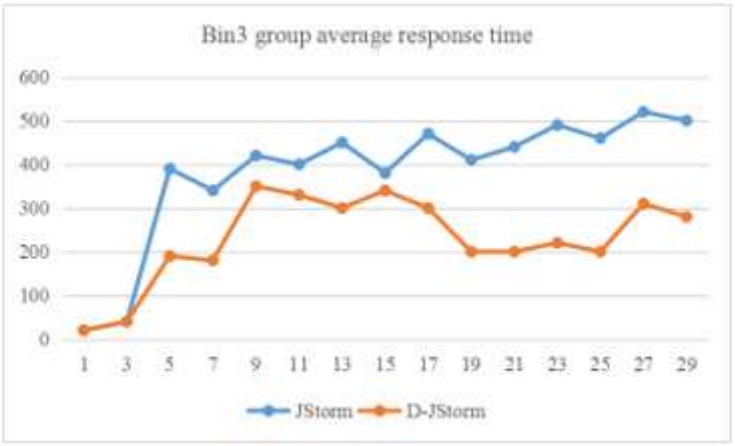


Figure 1

Comparison of the satisfaction of time-dependent demand under different data arrival strengths



(a) Average response time curve of Bin1 group (b) Average response time curve of Bin2 group



(c) Average response time curve of the Bin3 group

Figure 2

Comparison of average response time under different data arrival intensities

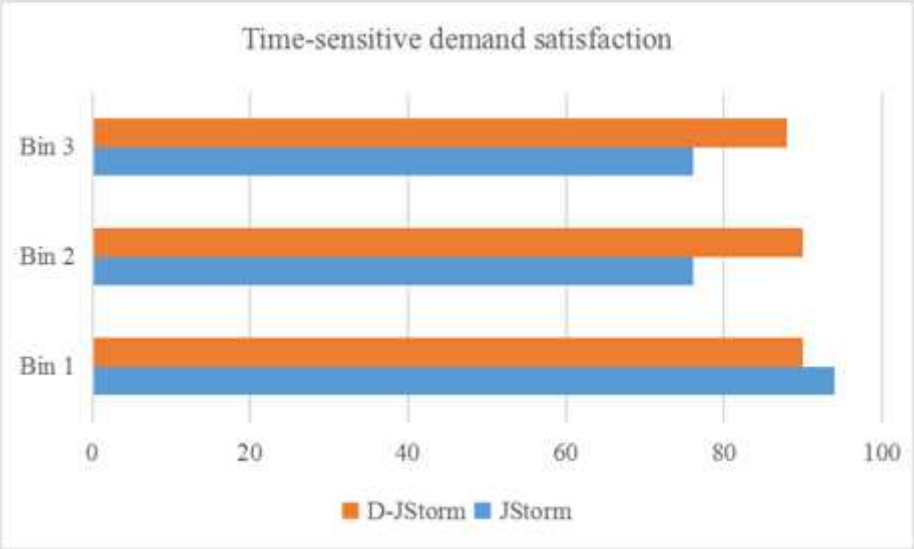
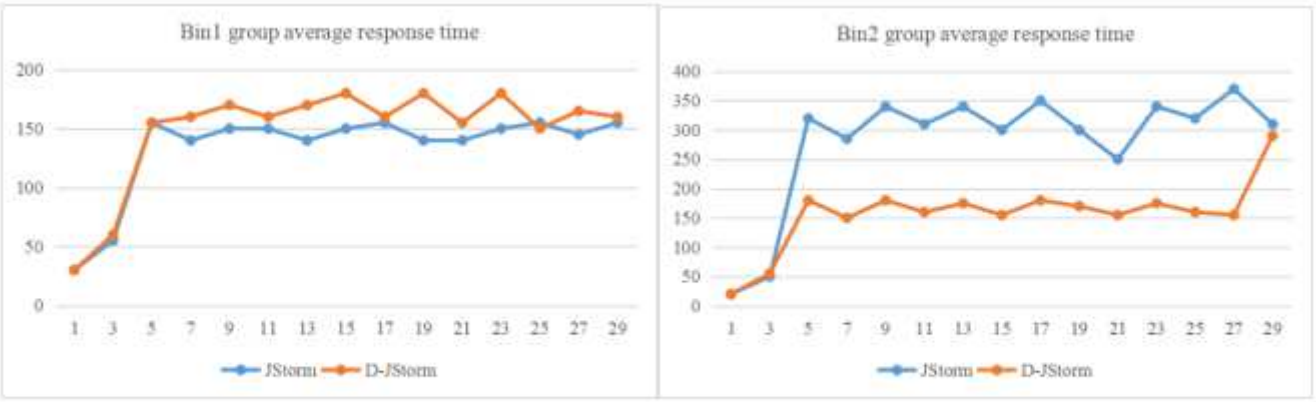
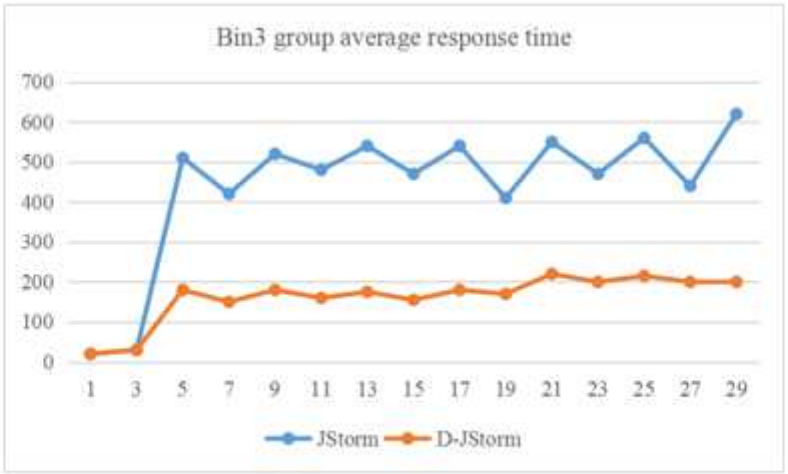


Figure 3

Comparison of time-based demand satisfaction degree under different initial resource configurations



(a) Average response time curve of Bin1 group (b) Average response time curve of Bin2 group



(c) Average response time curve of the Bin3 group

Figure 4

Average response time curve in different initial configurations

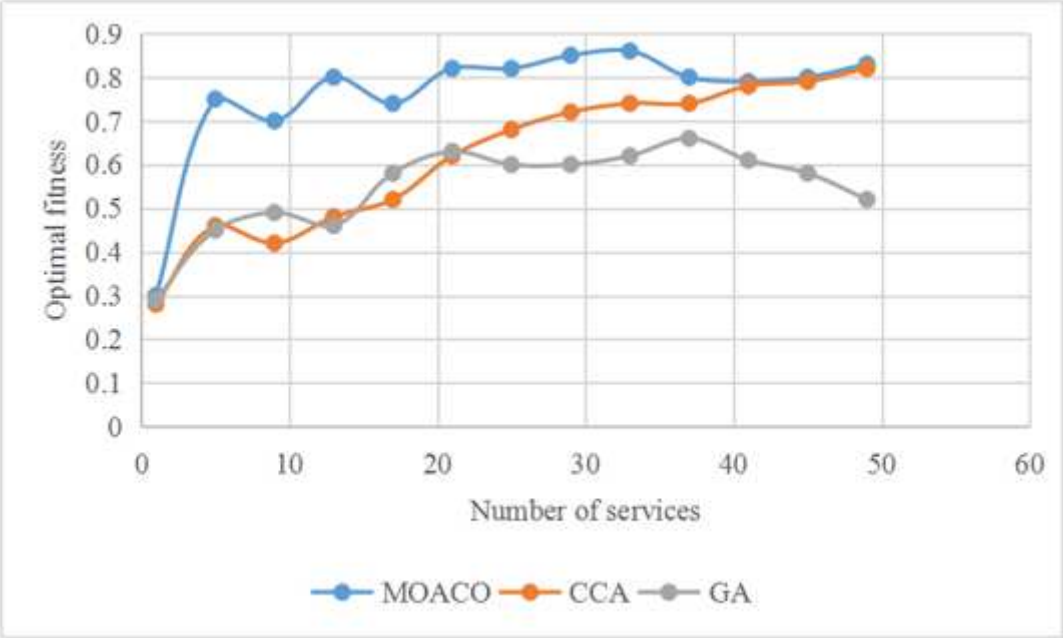


Figure 5

Optimal fitness at different service scales

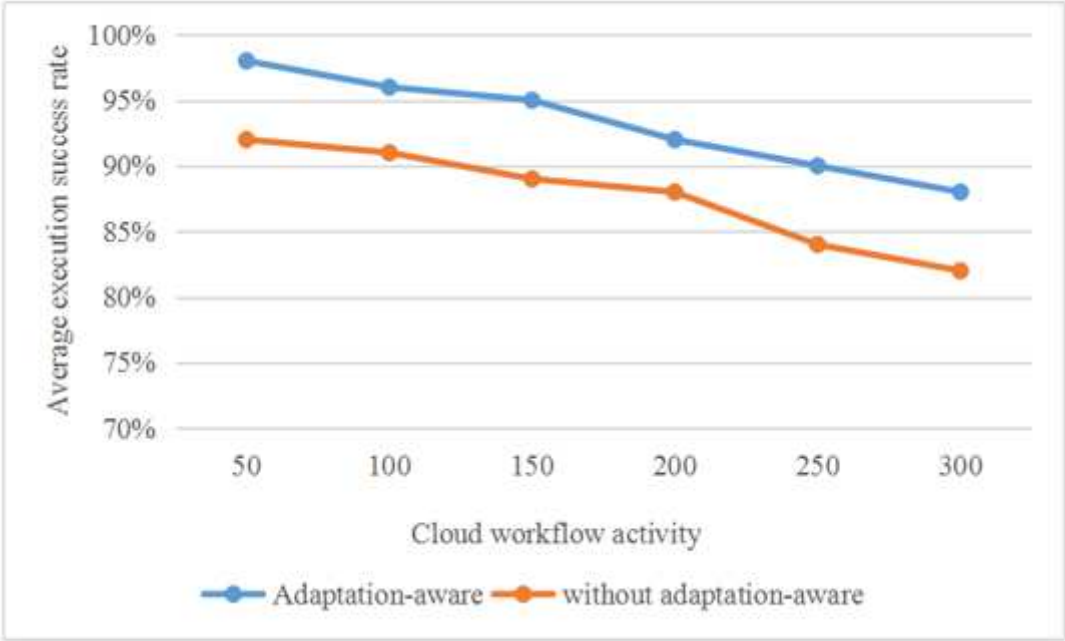


Figure 6

Effect of adjusting the perception policy on execution success rate

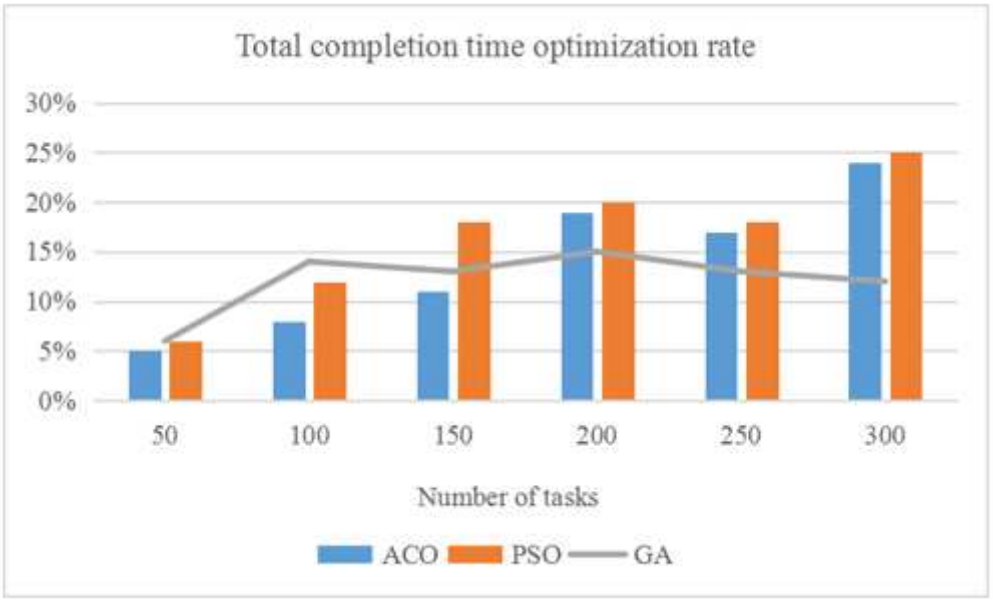


Figure 7

Optimization rate of overall completion time