Accepted Manuscript

Workflow scheduling applying adaptable and dynamic fragmentation (WSADF) based on runtime conditions in cloud computing

Zahra Momenzadeh, Faramarz Safi-Esfahani

PII: S0167-739X(18)30590-9
DOI: https://doi.org/10.1016/j.future.2018.07.041
Reference: FUTURE 4359

To appear in: *Future Generation Computer Systems*

Received date : 18 March 2018
Revised date : 16 May 2018
Accepted date : 17 July 2018

Please cite this article as:, Workflow scheduling applying adaptable and dynamic fragmentation (WSADF) based on runtime conditions in cloud computing, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.07.041

Workflow Scheduling Applying Adaptable and Dynamic Fragmentation (WSADF) Based on Runtime Conditions in Cloud Computing

Zahra Momenzadeh[1], Faramarz Safi-Esfahani[2] (Corresponding Author)
[1]Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran.
[2]Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran.

Abstract- **Workflows are a set of tasks and the dependency among them, which are divided into scientific and business categories. To avoid problems of centralized execution of workflows, they are broken into segments that is known as fragmentation. To fragment the workflow, it is highly important to consider the dependency among tasks and runtime conditions. The cooperation between the scheduler and fragmentor must be such that the latter generates appropriate tasks with optimized communication cost, delay time, response time, and throughput. To this end, in the present study, a framework is proposed for scheduling and fragmentation of tasks in scientific workflows that are conducted in fragmentation and scheduling phases. In the fragmentation phase, the fragments are generated with regard to the number of virtual machines available during runtime. In the scheduling phase, the virtual machines are selected with the aim of reducing bandwidth usage. The experiments are performed with three Configurations during both phases of fragmentation and scheduling. Response time, throughput, and cost (BW and RAM) were improved compared to the baseline studies on Sipht, Inspiral, Epigenomics, Montage, and CyberShake workflows as datasets.**

Key words: Scientific workflow, Scheduling, Dynamic and Adaptive Fragmentation, Adaptive Scheduling

1- Introduction

Scientific workflows might be very large, carrying a vast number of tasks and calculations, manipulate a huge amount of data and they will eventually be realized as thousands of concurrent process instances. Hence, the implementation of workflows in the cloud matters [1, 2]. The centralized execution of workflows increases response time and missing the deadline of a workflows, either. As a solution, distributed workflow engines come to action; therefore, workflows are fragmented into sub-workflows (fragments) so that they can be executed using distributed resources by a cloud scheduler. Fragmentation of workflows enhances scalability and reusability, as well.

Fragmentation is performed in dynamic and static modes. In the former, fragments are generated during the runtime, while in the latter, fragmentation occurs before the runtime. In static fragmentation, workflow is fragmented before the runtime and is then executed during the runtime by determining the resource. Although this method is simple, fragments are not compatible with the runtime conditions. Dynamic fragmentation decides on the generation and execution of fragments during the runtime. However, adaptable and dynamic fragmentation generates and executes the fragments based on the feedbacks from runtime environment in order to balance the scalability and efficiency of workflows. This approach is recommended when workflow engine acts as a cloud service to execute a huge number of workflow instances and/or when workflows include a large number of tasks [3].

So far, numerous studies have examined the fragmentation and scheduling of workflows. The FPD model [4] is a frequently used method in fragmentation, in which workflows are divided into single-task fragments, thereby increasing communication messages, delay time, and response time, and decreasing throughput. As a result, it is highly important to consider a method for reducing the number of fragments generated from a workflow. In the method proposed in [5], a method known as ATSDS was expressed for workflow fragmentation, which is a two-phase adaptive method for the fragmentation and scheduling of workflows during the runtime. The phases utilized in this method are fragmentation and resource allocation. Fragmentation is conducted based on the hierarchical process decentralization (HPD) algorithm that focuses on business processes and has no idea on scientific processes.

Scheduling is an important issue in workflow execution and many algorithms have been introduced for it. By far, the proposed scheduling methods consider limiting factors, e.g. task runtime and remaining time. The CTC [6], SLV [7], and QDA [8] algorithms are proposed for scheduling. These algorithms schedule and execute workflows based on deadline and with the aim of decreasing execution costs. The major problems with these algorithms include: 1) the method of mapping tasks to resource without considering the limiting factors such as runtime and deadline; 2) QDA, CTC, FPD, and SLV algorithms generate a large number of fragments; 3) communication messages among the generated fragments are increased that results in increasing bandwidth usage; 4) delay time is increased because the number of communication messages is increased, thereby increasing response time and decreasing throughput, either; 5) increasing the delay time results in missing the deadline before running all the fragments.

As a result, the main problem is adaptable and dynamic fragmentation and execution of scientific workflows based on the feedbacks from runtime environment. The first hypothesis states applying the number of virtual machines, and available bandwidth in dynamic fragmentation results in creating adaptive fragments that improve response time and throughput of a workflow engine at runtime. The second hypothesis says that the scheduler is able to reduce the cost of bandwidth usage, and memory usage of virtual machines, and satisfy the deadline of workflows by applying the parameter virtual machine cost along with the adaptive fragments. The main objective is to balance the scalability and efficiency of workflow engines as a service to execute scientific workflows. This research highlights the fragmentation and execution of scientific workflows in that 1) it does not provide a large number of fragments that is equal to high communication cost and delay; however, applies adaptive fragmentation based on runtime feedback; 2) it does not separate fragmentation from scheduling; however, schedules adaptive fragments based on the feedbacks from runtime environment. As the methodology of this research, in order execute scientific workflows adaptively, WSADF framework is presented that includes two phases of fragmentation and scheduling along with algorithms to support them. In the fragmentation phase, appropriate fragments are generated considering runtime conditions (i.e. the number of virtual machines, and available bandwidth among them). In the scheduling phase, the scheduler selects virtual machines for each fragment in order to reduce the cost of bandwidth usage, memory usage, and satisfying the deadline of workflows. Scientific workflows [9] including CyberShake, Sipht, Montage, Epigenomics, and Inspiral are used as datasets for evaluating the proposed framework and comparing it with other basic approaches.

The WorkflowSim [10] simulator is incorporated for the realization of idea. Various algorithms can be used for scheduling, but round-robin scheduling algorithm is used to start with. This algorithm does not require additional information on tasks (e.g. runtime and/or number of task instructions) and executes the tasks equally. Experiments were conducted in three configurations along with both phases of fragmentation and scheduling. Results were improved compared to the baseline studies. For instance, compared to Montage workflow in Configuration-1 and the fragmentation phase, there was 84.75% improvement in mean response time and 87.68% improvement in throughput. In Configuration-2 and the fragmentation phase, 84.64% improvement in mean response time and 83.46% improvement in throughput were observed. In Configuration-1 and the scheduler phase, it demonstrated 83.94%, 69.56%, and 96.91% improvement in mean response time, throughput, and bandwidth usage cost, respectively. In Configuration-2 and the scheduler phase, it demonstrated 94.49%, 47.82%, and 96.1% improvement in mean response time, throughput, and bandwidth usage cost, respectively. In Configuration-3 the results of experiments on datasets with variable deadline (625, 5000, 20,000 and 25,000) indicate that the success rate of the proposed framework was 100% than the base models.

This paper is organized as follows: Section 2 reviews the related studies. Section 3 presents the proposed method and Section 4 contains the evaluations. Finally, Section 5 includes the conclusions and suggestions for the future research.

2- Background and concepts

2-1- Workflow

Workflow means a group of tasks and their dependency. Workflows are divided into business and scientific categories: the former is a group of operations for performing a task, including two or more human factors, and the latter is discussed in medicine, meteorology, etc. Workflows differ based on their type (business or scientific) and type of dependencies among their tasks. If they are scientific, the dependency is data-driven, i.e. the next task uses the output data of the previous one. If they are business, the dependency is control-driven, i.e. the next task can be run only when the execution of the previous task is complete [11-18]. Workflows are among software applications utilized in cloud computing. Resource provision and task scheduling are required for running workflows in clouds. In the resource provision, the resource is of pay-per-use type, separated from task scheduling. In the task scheduling, the tasks are scheduled for utilizing the resources [3, 11-15, 19]. Scheduling, mapping task to resources, and running them are done based on priorities. Workflow scheduling is the allocation of tasks to resources, which may be distributed in numerous ranges. Scheduling is categorized into dynamic, static, and combined types [20].

2-2- Types of workflow fragmentation

Based on Fig 3, fragmentation is divided into two main groups: static and dynamic. If fragmentation is performed before runtime, it is called static. However, if it predicts which task must be placed in a fragment during runtime, it is called dynamic.

2-2-1. Horizontal, vertical, and diagonal fragmentation

The dependency among tasks in a workflow is one of the most important issues in fragmentation. Numerous reasons exist for the fragmentation of workflows. Reducing the cost and time of communication by adapting fragmentation to data distribution, adapting fragments to organizational structures and resources, resolving hardware limitations, and regulating efficiency by workflow distribution are the motivations for workflow fragmentation. Workflow fragmentation is divided into horizontal, vertical, diagonal, and combined categories. A workflow is decomposed into two horizontal fragments if they are not dependent on each other. Alternative or concurrent processes can be fragmented horizontally (Fig 1 (a, b)). A workflow is decomposed into two vertical fragments if one fragment depends on the other and there is no dependency in the opposite direction. If a fragment includes exactly one proposition, it is called orthogonal, which is a type of vertical fragmentation. Fig 2(a) and (b) illustrates vertical and orthogonal fragmentations, respectively [21].

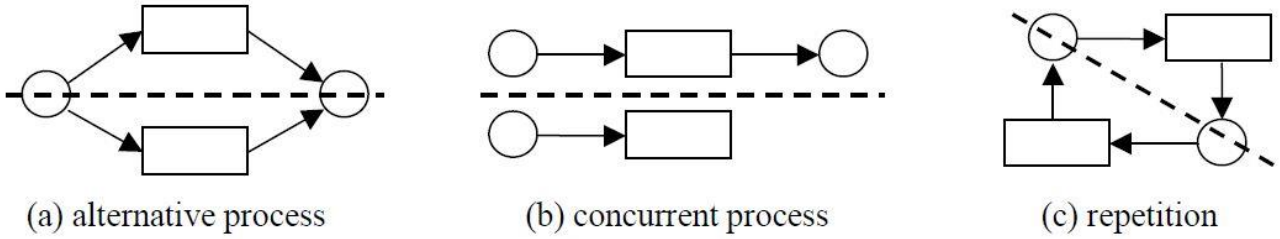


Fig 1: Horizontal process patterns [21]

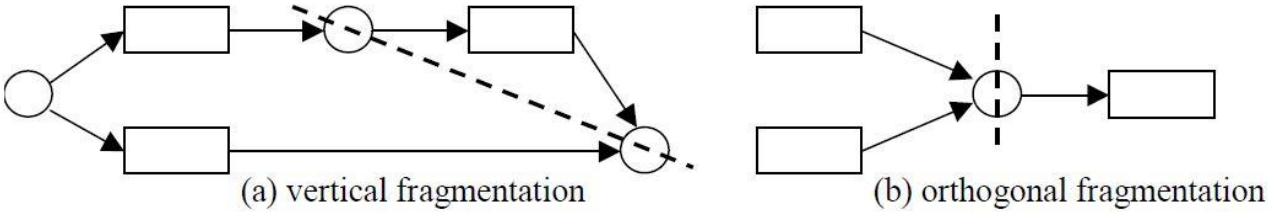


Fig 2: Vertical Process patterns [21]

If the tasks in a workflow have circular dependency, they are fragmented diagonally. Fig 1(c) presents diagonal fragmentation. When a combination of the noted methods is used for the fragmentation of a workflow, it is called combined fragmentation [21].

3- Related Works

In this section, we review the previous studies on scheduling and fragmentation; although several research papers [22-28] study different aspects of cloud computing. The studies are compared with the proposed method in terms of fragmentation and scheduling based on their objectives.

According to Fig 3, workflow fragmentation is performed in static and dynamic forms, each with various categorizations. The general static and dynamic states are divided into horizontal, vertical, diagonal, and combined categories. While, in the present study, combined dynamic fragmentation and dynamic scheduling were incorporated. Fig 4 illustrates important workflow problems from a different points of view.

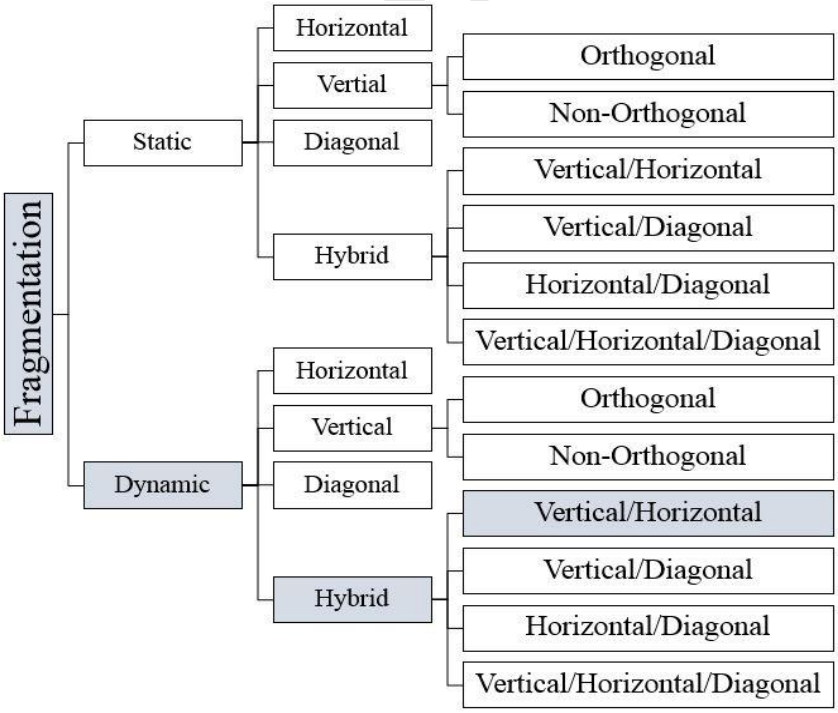


Fig 3: Mind Map of workflow fragmentation

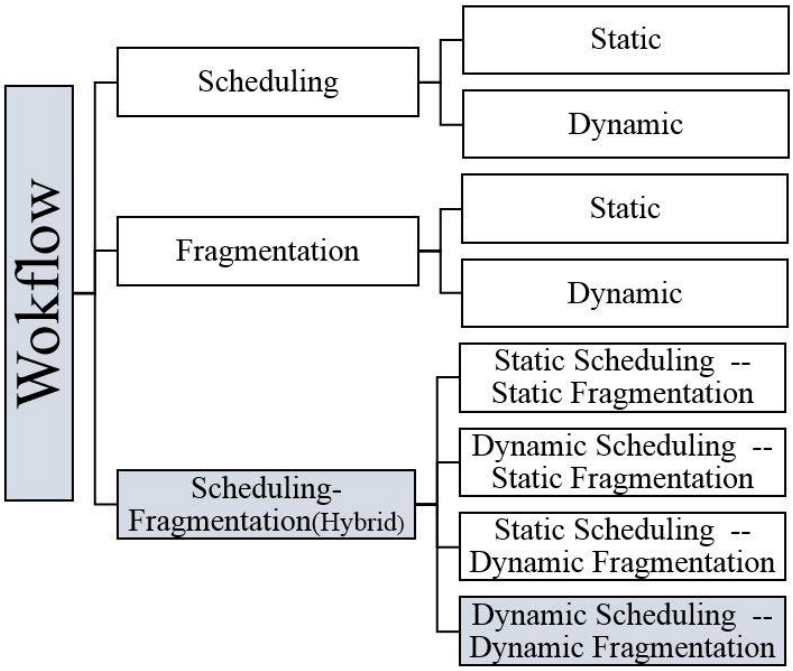


Fig 4: Mind Map of Workflows in terms of Scheduling and Fragmentation

3-1- Workflow Fragmentation Studies

In this section, we review previous studies on workflow fragmentation. Afterwards, each study is compared with the proposed WSADF framework according to Table 1. Wei Tan et al. [29] employed a centralized workflow known as CWF-net for workflow fragmentation. In this method, tasks or operations are considered as the transition and the dependency among them is regarded as the place. Also, the fragmentation method is static. The disadvantages of this method include fragmentation before execution, which reduces efficiency and flexibility. This method differs from the present framework in terms of the selection of tasks and the selected fragmentation method. Wei Tan et al. [30] proposed a method for dynamic workflow fragmentation in distributed environments. By partitioning centralized workflow into segments, each segment can be transferred to other servers to be executed. The advantage of dynamic fragmentation is in enhancing the flexibility and efficiency of systems and preventing the transfer of duplicate information. The difference between this study and our proposed framework is in the selection of tasks to be placed as one segment. Guoli Li et al. [4] proposed the FPD method for workflow fragmentation, in which a workflow was completely divided into single-task fragments. Then, each fragment was given to the scheduler to receive resources and the fragmentation method is static.
The mentioned method differs from the present framework in terms of fragmentation method. The disadvantages of this study include the static nature of work, which reduces efficiency and flexibility in resource selection as well as the task selection method. Peter Muth et al. [31] proposed the centralized method for workflow fragmentation. In the centralized fragmentation, each workflow is given to the scheduler for resource mapping and execution in a centralized manner, without being divided into small fragments. In the centralized state, the number of generated fragments equals 1. For centralized workflows, there is a standard known as Workflow Management Coalition. It differs from the present framework in terms of fragmentation and execution of fragments as well as dynamic execution and fragmentation. Khorsand et al. [5] introduced a method known as ATSDS which was a two-phase adaptive method for workflow fragmentation and scheduling during runtime. The phases utilized in this method are fragmentation and resource allocation. In the fragmentation phase, the fragmentor divides

business workflows into appropriate fragments using three algorithms and considering the parameters of the number of virtual machines and mean bandwidth. In the first algorithm, appropriate fragments are generated considering the number of virtual machines and using HPD. Then, a suitable level of granularity is achieved considering bandwidth and using fuzzy logic. This method differs from the present framework in terms of workflow type, fragmentation method, selection of fragments, considering fuzzy logic, and resource allocation to each fragment. Khaled Almi Ani et al. [32] introduced a model entitled PBWS for partitioning business workflows. This method consists of three major steps: partitioning, partition regulation, and resource allocation and is different from the present framework in terms of fragmentation and scheduling methods. Sherry X Sun et al. [33] proposed a framework based on process extraction for business workflow fragmentation. This framework consists of four phases. The first phase is saving the logs. When a workflow is running, the log of each task is stored in a database. The second phase is the extraction model for centralized workflows. In this phase, when the workflow enters, the process extraction algorithm attempts to discover the model describing the sequences of tasks. The third phase is the time analysis of centralized workload properties. The fourth phase is workflow fragmentation and distribution. In this phase, the tasks are selected for fragments in order to achieve minimum time. This framework is different from the present framework in terms of task selection and fragment generation method. The type of the selected workflow differs as well. Previous algorithms are compared with the proposed framework in terms of fragmentation with various factors in Table 1.

Table 1: Comparison of fragmentation algorithms with the proposed WSADF framework

Algorithm or method	Fragmentation method	Reducing runtime	Reducing response time	Guaranteed execution of the fragment	Optimizing the use of resources	Compatibility with runtime conditions	Improved throughput	Reduce bandwidth cost
Wei tan et al[30]	Dynamic	-	×	-	✓	✓	×	×
Wei tan et al[29]	static	-	×	✓	×	×	×	×
Guoli Li et al[4](FPD)	static	✓	×	-	×	×	✓	×
Peter Muth et al[31](Centralized)	-	-	✓	×	×	×	×	×
Khorsand et al[5](ATSDS)	Dynamic	-	-	-	✓	✓	-	✓
Khaled Almi et al[32] (PBWS)	Dynamic	✓	-	-	✓	-	-	-
Sherry X,sun et al[33]	static	✓	-	-	-	-	-	-
WSADF (Presented Framework)	Dynamic	-	✓	✓	✓	✓	✓	✓

3-2- Workflow Scheduling Approaches based on Fragmentation

In this section, we review the previous studies on business workflow scheduling. Afterwards, each study is compared with the proposed WSADF framework according to Table 2**.** Faisstnauer et al. [34] introduced a technique for enhancing the round-robin scheduling. By adding priorities to tasks, this technique prevents starvation and enhances efficiency. The prioritizing level of each task is based on the error factors determined by the user. Tasks with larger error factors must be executed first. In the proposed framework, workflow is fragmented dynamically, ignores runtime conditions, and is finally

executed. Oprescu et al. [35] proposed a scheduling algorithm. Scheduling is performed based on budget limits. Based on the BaTs method, tasks are categorized into Bag of tasks and are then scheduled and executed using the round-robin algorithm. This technique differs from the present framework in terms of categorizing tasks and scheduling methods. In the fragmentation method introduced by Muthusamy et al. [7], workflow is completely fragmented such that there exits one task per fragment. The major objective of this model is scheduling with the lowest cost (SLV). In this paper, fragmentation was performed statically, while scheduling was done dynamically. The difference is in terms of workflow fragmentation and scheduling method. Bansal et al. [36] proposed a heuristic scheduling algorithm which can dynamically adapt and schedule tasks without a priori information. Two queues exist in this method, one for tasks waiting to be executed and the other for tasks which are being executed in at least one machine. Tasks waiting to be executed are prioritized and executed using the round-robin algorithm based on the costs of task execution. This method and the present method differ in terms of task fragmentation and prioritizing methods. Abrishami et al. [37] proposed an algorithm based on service quality according to minor critical path. The aim of this algorithm was to minimize workflow execution costs before reaching the deadline. The difference between this algorithm and the present framework lies in the fact that the latter uses robin-round algorithm for task scheduling and dynamically performs workflow fragmentation considering runtime conditions, leading to higher adaptability with resources and optimal resource use. Abba et al. [38] proposed three scheduling algorithms. In the first algorithm, EPFRR, the task governing the deadline has the highest priority for scheduling and erection. Here, tasks are prioritized in an ascending order based on the deadline. In the second algorithm, LSTRR, tasks are ordered based on the shortest remaining time until the completion of execution, identified using round-robin algorithm in quantum, and exclusively executed. In the third algorithm, SPTFRR, the tasks are ordered based on the shortest process time to the system, exclusively identified using round-robin algorithm in quantum, and executed by the system. Donyadari et al. [11] introduced a method for scheduling based on round-robin algorithm. In this method, the tasks in workflow are first ordered in queue using the parallel depth search algorithm. Afterwards, the degree of dependency and deadline of each task is determined and they are prioritized in an ascending order based on the deadline. Then, scheduling and execution are performed based on the closest deadline. The major difference between this method and the present framework is that the workflow fragmentation model is dynamic and scheduling of tasks is performed considering bandwidth usage cost reduction. Ke Liu et al. [6] introduced a method for business workflow scheduling called CTC. In this method, the workflow is first divided into single-member fragments and each fragment receives a deadline. Later, runtime is estimated, costs are calculated, and each fragment is allocated the services it demands. Here, the aim of scheduling is to reduce resource use costs and achieve the shortest runtime. This method and the present method differ in terms of fragmentation and scheduling methods. Huifang Li et al. [8] introduced a method for business workflow fragmentation known as QDA. Similar to CTC, this method completely fragments the workflow. A deadline is considered for each fragment. Subsequently, low-cost services are selected for scheduling and execution. During execution, if cheaper services are found, they are used. The difference between this method and the present framework is similar to the difference mentioned for CTC. Previous algorithms are compared with the proposed framework in terms of scheduling with various factors in Table 2.

Table 2: Scheduling algorithms compared with the WSADF

Algorithm or method	Reducing runtime	Optimizing the use of resources	Compatibility with runtime conditions	Improved throughput	Reduce cost
Faisstnauer et al[34]	×	✓	×	×	×
Oprescu et al[35]	✓	×	✓	×	×

Bansal et al[36]	✓	×	✓	×	×
Abrishami et al[37]	✓	×	×	×	✓
Abba et al[38]	✓	×	×	×	×
Donyadari et al[11]	✓	✓	×	×	×
Muthusamy et al [7](SLV)	×	×	×	×	✓
Ke et al [6](CTC)	✓	×	×	×	✓
Li et al[8](QDA)	✓	×	×	×	✓
WSADF (Presented Framework)	-	✓	✓	✓	✓

3-3- The comparison of WSADF with the baseline studies

Table 3 shows the features of the proposed framework and some of the baseline studies.

Table 3:Specifications of proposed framework and some of the compared algorithms

Title	Fragmentation Algorithm	Scheduling Algorithm	Evaluation Criteria	Environment
FPD[4]	FPD	Round-Robin	Throughput, Average Execution Time	Cloud
SLV[7]	FPD	Round-Robin	Cost	Cloud
CTC[6]	FPD	FCFS	Execution Cost	Cloud
QDA[8]	FPD	MAX-MIN	Cost	Cloud
WSADF (Presented Framework)	WSADF-Fragmentation Algorithm	WSADF-Scheduling Algorithm	Bandwidth Response Time, Throughput	Cloud

4- WSADF: Workflow Scheduling Applying Adaptable and Dynamic Fragmentation

The task selection method for generating fragments is very important for the fragmentation of workflows. In the FPD [4] fragmentation algorithm, communication messages among fragments are increased because as many fragments as tasks are generated in a workflow. This issue increases delay time, thereby increasing response time and reducing throughput. In the method proposed in [5], ATSDS is expressed for workflow fragmentation. This method is a two-phase adaptive method for the fragmentation and scheduling of workflows during runtime. The phases utilized in this method are fragmentation and resource allocation. Fragmentation is conducted based on the HPD algorithm this method is proposed for the fragmentation of business workflows. Also, methods are introduced for scheduling in CTC [6], SLV [7], and QDA [8] algorithms. In these methods, the reduction of bandwidth cost and used memory is taken into account. In the present study, WSADF framework is presented which resolves the problems of these algorithms. WSADF framework comprises a fragmentation algorithm, a scheduling algorithm, a fragment repository, and a scheduler as illustrated in Fig 5. The input of fragmentation algorithm is a workflow and its output is the fragments generated from the workflow. Fragment repository is a repository for storing workflow and its' fragments. The next phase is the scheduling algorithm that takes tasks from the fragment repository and schedule them. Based on the least bandwidth usage cost, the scheduler allocates virtual machines to each fragment. The fragments are then executed. There is also another repository for the runtime data, as shown in Fig 5.

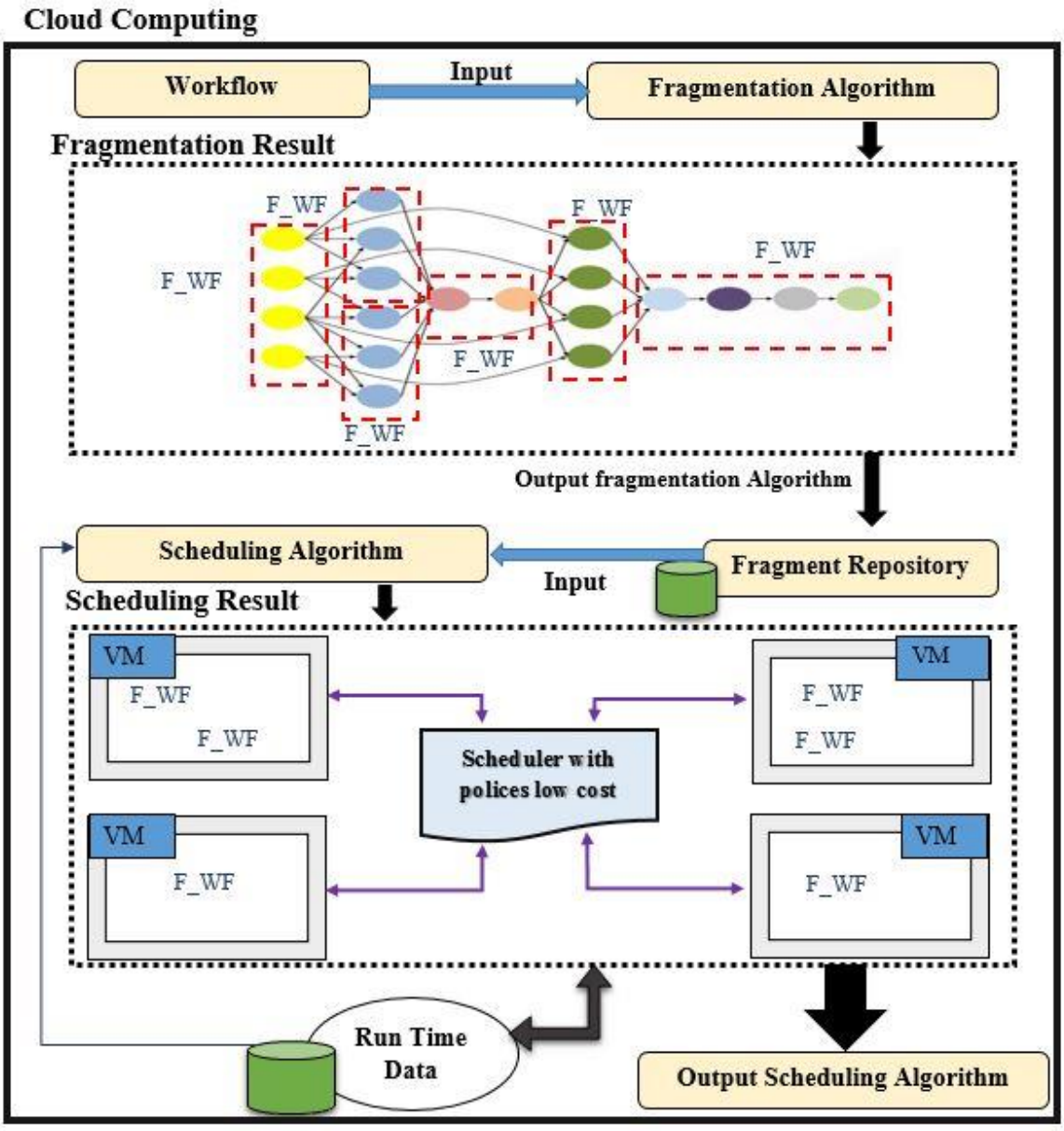


Fig 5:Framework of Workflow Scheduling Applying Adaptable and Dynamic Fragmentation (WSADF)

4-1- Relationship of WSADF components:

This section describes the relationship of WSADF framework components as shown in Fig 6. Accordingly, the fragmentation phase receives a workflow and a number of virtual machines. Then, the fragmentor generates a fragment based on the current conditions and sends it to the scheduling phase. The scheduler selects a virtual machine for each fragment so that reduces costs and the fragment is executed, subsequently. This cycle is repeated until no task remains in the workflow for fragment generation.

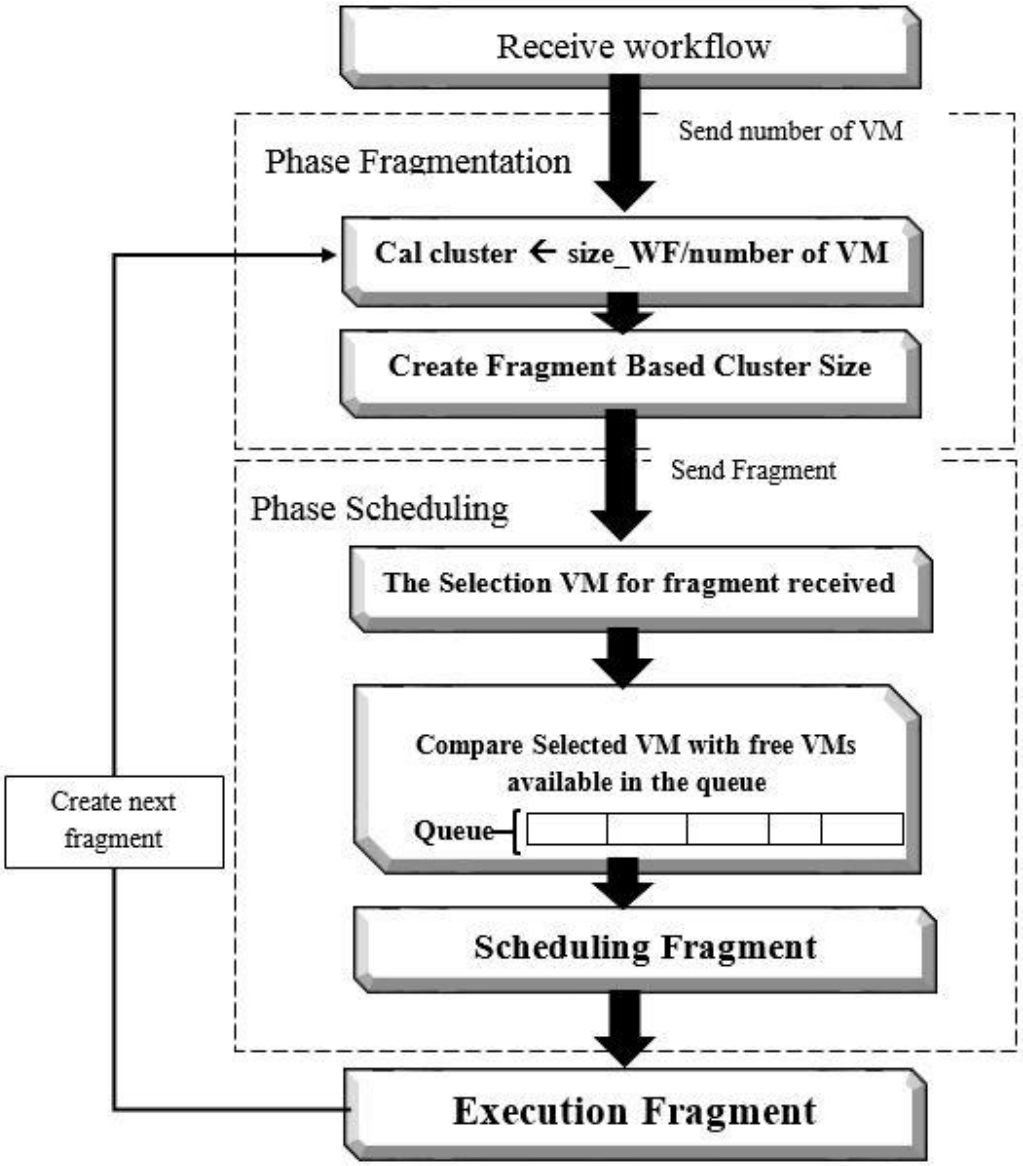


Fig 6: Relationship of WSADF framework components

4-2- Algorithms of WSADF framework

This section illustrates flowchart and algorithms of WSADF framework. Fig 7 demonstrates the flowchart of the fragmentor phase. This flowchart is started when a workflow enters. After examining runtime conditions, the workflow is fragmented in terms of the number of virtual machines. The resulting value (cluster) denotes the number of tasks, which can be placed in one fragment. Afterwards, by examining the cluster>0 condition, a task is selected to be placed in one fragment. When this cycle is over, each generated fragment is sent to the scheduler to receive a resource to be executed. This cycle continues as long as there are tasks for fragmentation in the workflow. At the end of the execution, a list of scheduled and executed tasks is printed. Fig 8 shows the flowchart of the scheduling phase of WSADF. After receiving a fragment, the scheduler selects one virtual machine from the list of virtual machines and allocates it to the fragment. Then, it checks whether another (free) resource with less cost is available on the list. If there is a resource with less cost, the current resource is replaced with the new one.

Otherwise, the task will continue using the current resource. To clarify this issue, pseudo-codes of the fragmentation and scheduling algorithms are presented in Fig 9 and Fig 10, respectively.

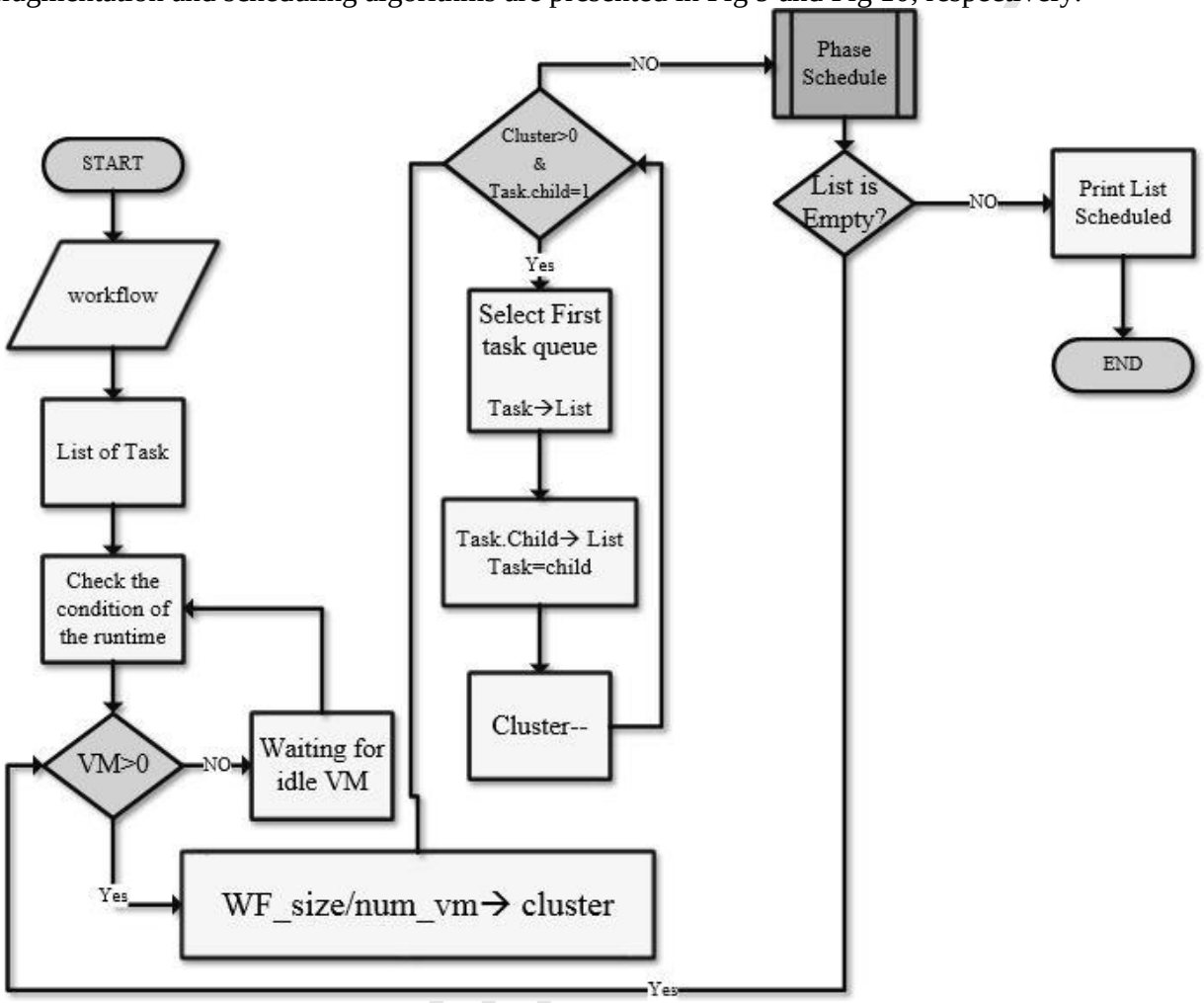


Fig 7: fragmentation flowchart

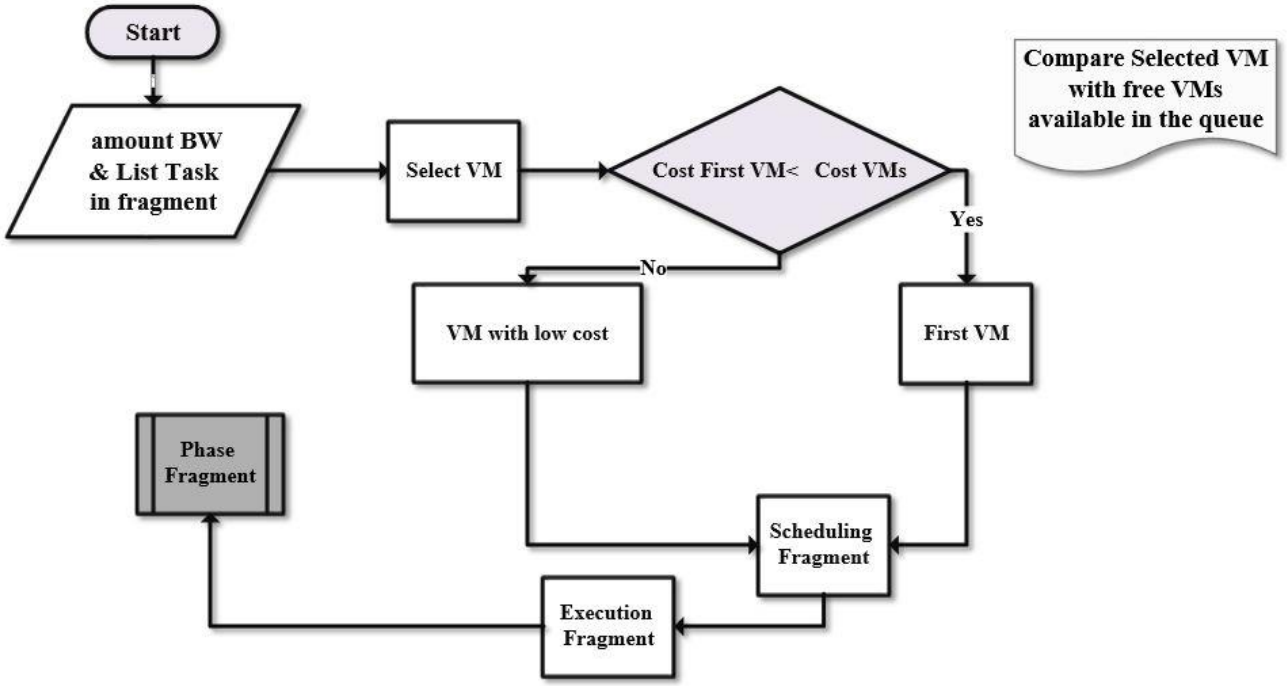


Fig 8: scheduling flowchart

Algorithm 1: Fragmentation

```
1.  Input workflow W(T,E) for the fragmentation phase
2.  Output fragment from workflow in the fragmentation phase
3.  Fragmentation:
4.  Num=workflow_size/num_vm //Calculate the number of tasks in the fragment
5.  List is Empty
6.    Loop Num>0 or Task.child==1
7.  Task=List.First.Task;
8.  Fragment.Put(Task);
9.  If(Task.child==1)
10.     Task=Task.child;
11. Else
12.     Task=List.next.Task;
13. Create Fragment
14. Send fragment to schedule
```

Fig 9: Pseudo-code fragmentation algorithm in WSADF framework

Algorithm 2: Scheduling

```
1.  Input fragment for Scheduling phase
2.  Output List from Task Scheduled in Scheduling Phase
3.  Schedule:
4.  Select VM free from list
5.  If(VM Low cost BW)
6.    Continue Scheduling;
7.  Else
8.     Search VM with lower Cost of BW
9.  If(VM is busy)
10.    Continue Search VM with lower Cost of BW
11. Else
12.     Select VM
13.     Execute Task
14.     Next Task from List
15. Next fragment (send request fragment to phase fragmentation)
```

Fig 10: Pseudo-code of scheduling algorithm in WSADF framework

4-3- Case study: An example of the fragmentation and scheduling algorithms in WSADF

This section presents an example to examine WSADF framework. The Montage workflow [9] is a scientific workflow created in the astronomy field with the aim of creating mosaics from Sky. This workflow is generated by processing, small images of sky, and its size depends on the number of processed images. It is assumed that Montage workflow with 25 tasks is the input of the fragmentation phase, and five virtual machines are considered as resources. Based on Equation 1, the number of tasks placed in one fragment is calculated based on the number of available virtual machines. In this example, the value calculated using Equation 1 equals 5. Therefore, 5 tasks can be allocated to each fragment based on the number of resources (t_0, t_1, t_2, t_3, t_4). Afterwards, tasks are selected based on the parent-child dependency and the calculated value. The first fragment is selected along with 5 tasks to be submitted to the scheduling phase, and then a virtual machine is allocated to it. Subsequently, it checks whether this virtual machine has the lowest bandwidth usage cost. According to Fig 12, the virtual machine VM_0 is allocated and executed. The next fragment is generated with 5 tasks by the fragmentation phase based on the conditions used for the generation of the first fragment (t_5, t_6, t_7, t_8, t_9). Afterwards, in the scheduling phase, a virtual machine is allocated to it. The second fragment receives virtual machine VM_3 for execution. Based on the number of allowable tasks (i.e. 5) and the available parent-child dependencies, the third fragment cannot accept more than 4 tasks (t_{10}, t_{11}, t_{12}, t_{13}). Subsequently, it is sent to the scheduling phase to receive a virtual machine. First, virtual machine VM1 is allocated to it. Then, the bandwidth cost is examined that reveals there is a virtual machine with less cost. Therefore, instead of VM_1, VM_2 is allocated to the third fragment. The generation and scheduling of all the fragments continue in this way until the workflow has no task left for fragmentation and scheduling. Figs 11 and 12 depict the performance of this framework.

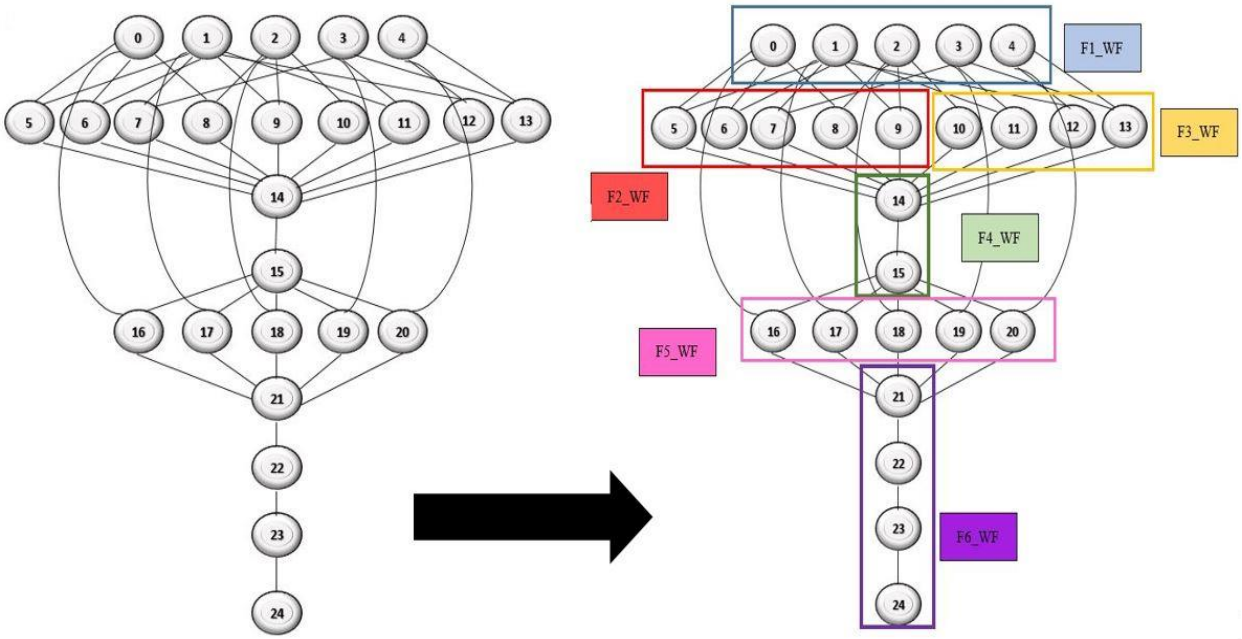


Fig11: Montage workflow with 25 tasks on the left -Workflow after fragmentation on the right

(Sample fragmentation with WSADF)

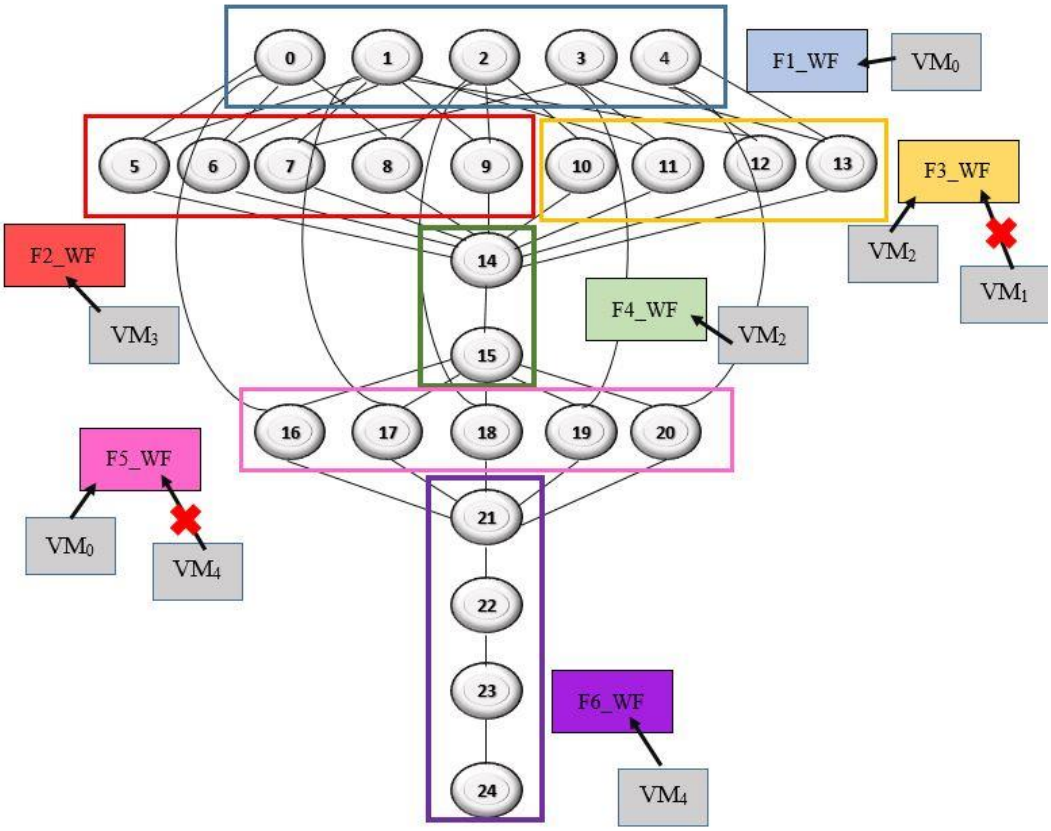


Fig 12: Schedule, workflow montage with 25 tasks and 5 virtual machines (sample scheduling with WSADF)

5- Evaluation

The proposed idea is evaluated in both scheduler and fragmentor phases. Five experiments are designed with three Configurations that are shown in Table 4. The WSDAF framework is compared with the FPD algorithm in the fragmentation phase, and with the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms in the scheduling phase. Five standard scientific workflows including Epigenomics, Montage, Sipht, CyberShake, and Inspiral are used as datasets in the experiments [9]. Fig 13 illustrates the relationship between WSADF framework and WorkflowSim [10]. By presenting a higher level of workflow management, WorkflowSim is introduced by expanding the CloudSim simulator [39]. The framework of WorkflowSim has been expanded from the core of CloudSim and its supporting programming language is Java. WorkflowSim can be used in programming environments which support Java [10]. We only show the experiments related to Montage for the sake of brevity.

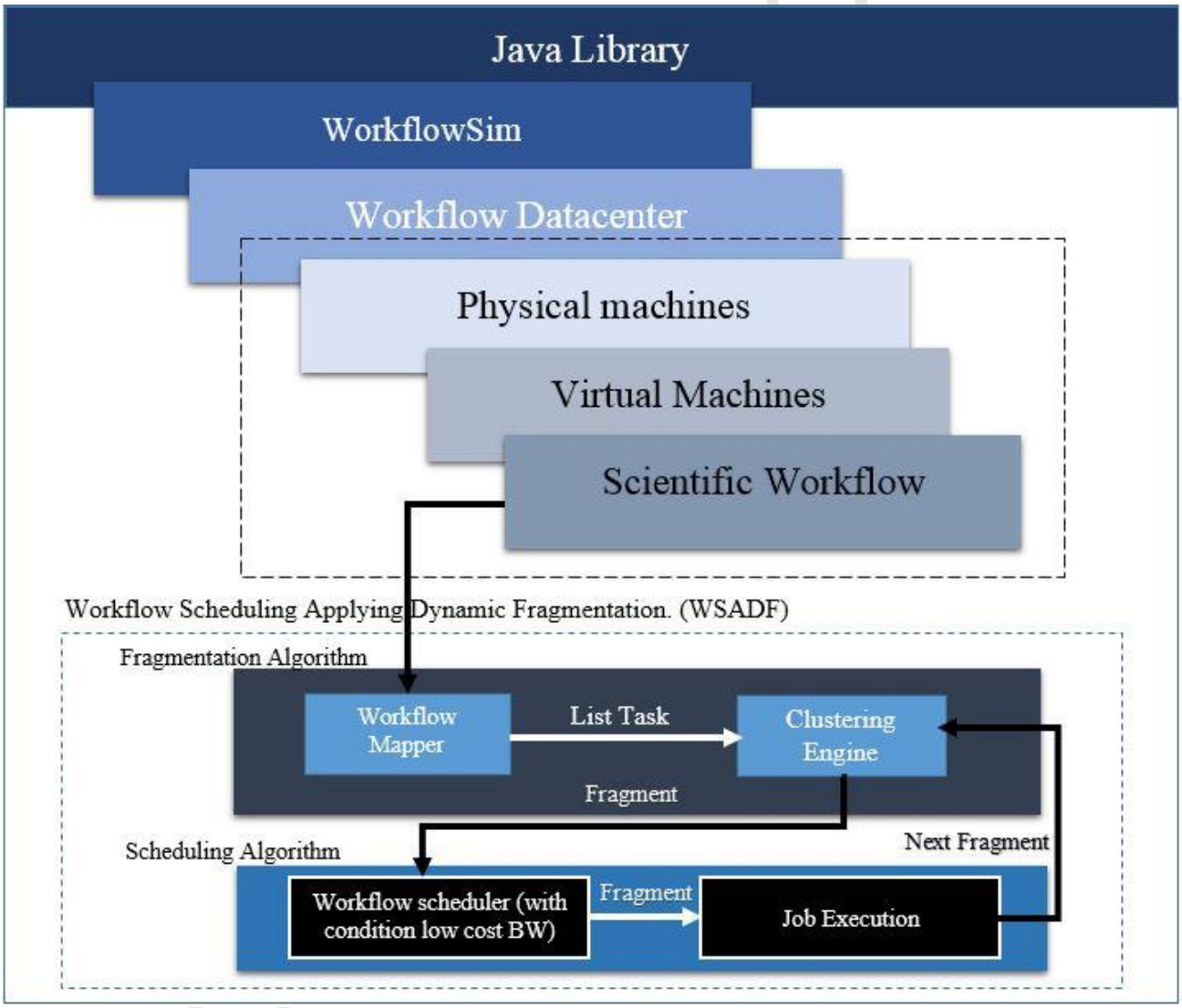


Fig13: the relationship of WSADF framework and Workflowsim

Table 4: Specifications of experiments (MS: Message size, VM: virtual machine, DL: deadline)

Experiment	phase	Configuration1			Configuration2			Configuration3		
		MS	VM	DL	MS	VM	DL	MS	VM	DL
Experiment1	Fragmentation	Static	Variable	…	…	…	…	…	…	…
Experiment2	Fragmentation	…	…	…	Static	Variable	…	…	…	…
Experiment3	Scheduling	Static	Variable	…	…	…	…	…	…	…
Experiment4	Scheduling	…	…	…	Static	Variable	…	…	…	…
Experiment5	Fragmentation/ Scheduling	…	…	…	…	…	…	Static	Static	Variable

5-1- Evaluation criteria

The proposed framework focuses on undirected acyclic graphs that are scientific workflows. These workflows include tasks and the relationships among them. The present study incorporated virtual machines as the resource that are different in terms of bandwidth and memory costs. Table 5 shows the equations employed in this study. Equations 1, 2, and 3 are used to calculate the number of tasks in each fragment, response time, and mean response time, respectively. Equation 4 is utilized to evaluate throughput percentage. Equation 5 is incorporated to calculate bandwidth costs. Equation 6 is used to calculate the cost of the used memory (MB). To calculate mean response time improvement and throughput improvement, Equations 7 and 8 are used.

Table 5:The formulas used in this study

Number	Reference	Formula
1	This Paper	$cluster = \frac{Workflow\ size}{Number\ of\ VMs}$
2	[1]	$Response\ Time = \sum_{i=0}^{n} Execution\_StartTime$
3	[1]	$AVG\ Response\ Time = \frac{Response\ Time}{num_{Task}}$
4	[1]	$Throughput = \frac{RequestNUM}{FinishTime} * 100$
5	[7]	$CostPerBW = \sum_{i=0}^{n} (CostPerBW\ of\ VM * amount\ current\ allocated\ BW)$
6	[7]	$CostPerRam = \sum_{i=0}^{n} (CostPerRam\ of\ VM * amount\ current\ allocated\ Ram)$
7	[1]	$Improvment_{RT} = \frac{AVGRT_{othermethod} - AVGRT_{WSADF}}{AVGRT_{othermethod}}$
8	[1]	$Improvment_{TP} = TP_{WSADF} - TP_{othermethod}$

5-2- Configuration of experiments

Here, evaluation is performed in three Configuration. The features of these Configuration are depicted in Table 6. These tables show the Configuration of the simulator for executing WSADF framework and baseline studies. In Configuration-1, the number of virtual machines is considered variable, while the size of messages is considered constant. In Configuration-2, the number of virtual machines is considered

constant, while the size of messages is considered variable. In Configuration-3, the number of virtual machines is considered constant, the message size is considered constant, and the deadline is considered variable.

Table 6: Three Configuration Of the experiments

	Settings1				Settings2				Settings3			
Number of Datacenter	1				1				1			
Number of Virtual Machine	**5**	**15**	**20**	**25**	20				20			
Message Size	500				**0**	**100**	**250**	**500**	500			
Bandwidth	1024				1024				1024			
Number of Task	30 or 25	50 or 60	100	1000 or 997	30 or 25	50 or 60	100	1000 or 997	30 or 25	50 or 60	100	1000 or 997
Deadline	-----				-----				**625**	**5000**	**10000**	**25000**

5-3- Evaluation of fragmentation phase

5-3-1. Experiment-1: Fragmentation using Configuration-1 (variable number of virtual machines; constant message size)

The fragmentation phase with Configuration-1 decreases mean runtime and increases throughput, while message size is assumed constant and the number of virtual machines is assumed variable. The following Fig14 shows the results of Experiment 1 on the proposed framework and FPD [4] algorithm with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean response time. Based on Fig 14, by increasing the number of tasks in the workflow, WSADF framework has much better results than FPD algorithm in terms of mean response time.

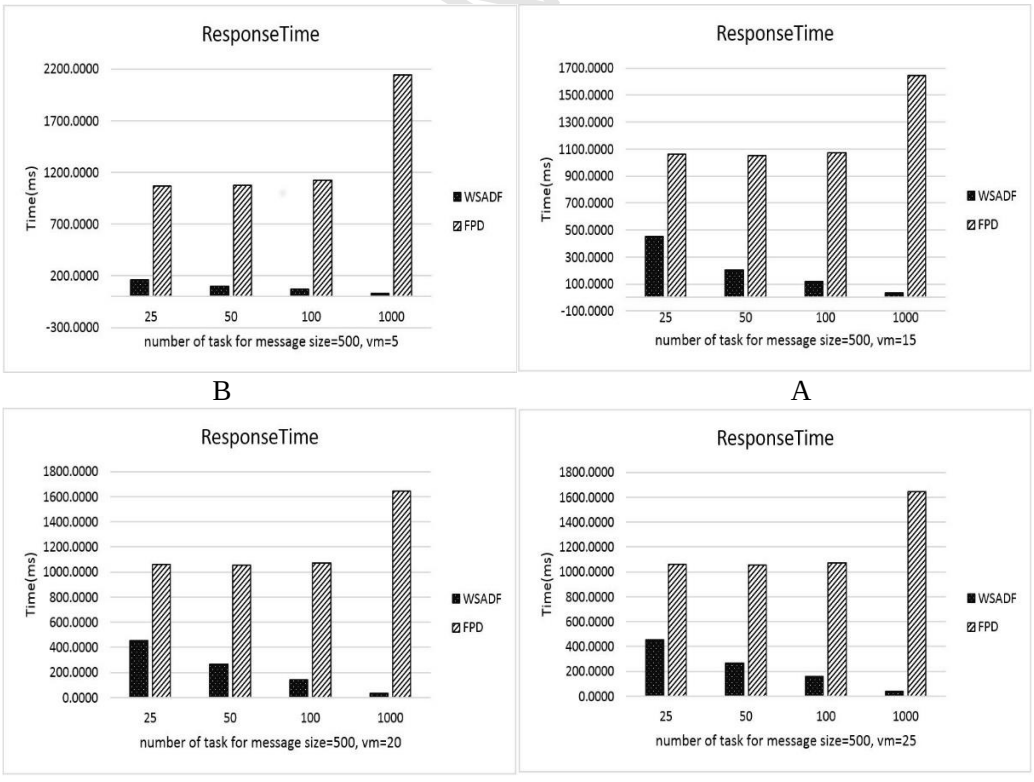


Fig 14: (A-D)The results of the mean response time from the experiment 1

The following Fig15 shows the results of Experiment 1 on the proposed framework and the FPD [4] algorithm with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Throughput. Based on Fig 15, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the FPD algorithm in terms of mean Throughput.

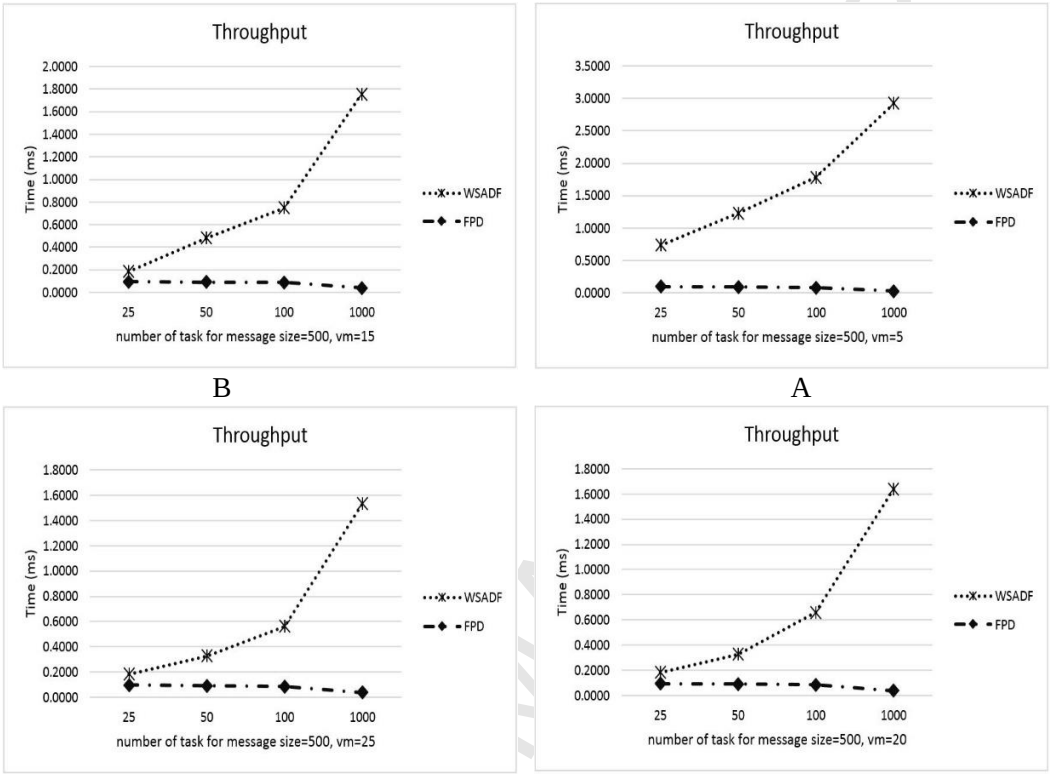


Fig 15 :(A-D) the results of the mean Throughput from the Experiment-1

Experiment 1 examines the fragmentation phase with Configuration-1 and Montage (Mon), Epigenomics (EPI), Inspiral (INS), CyberShake (CYB), and Sipht (SIP) scientific workflows as datasets. Results of the improvement of the proposed framework are compared with those of FPD [4] algorithm in Table 7. Results show the improved performance of the proposed framework compared to FPD algorithm since the former controls fragment generation by considering the number of virtual machines variable and message size constant.

Table 7: Results of the improvement of WSADF framework than the FPD (Fragmentation-Configuration-1)

Fragmentation – Configuration 1										
	Response Time					Throughput				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	**84.75**	**94.39**	**84.74**	**82**	**91.45**	**87.68**	**1.16**	**7.15**	**52.18**	**6.03**

5-3-2. Experiment-2: Fragmentation using Configuration-2 (constant number of virtual machines; variable message size)

Fragmentation phase with Configuration-2 reduces mean runtime and increases throughput, while message size is assumed variable and the number of virtual machines is assumed constant. The following Fig16 shows the results of Experiment 2 on the proposed framework and the FPD [4] algorithm with

Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean response time. Based on Fig 16, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the FPD algorithm in terms of mean response time.

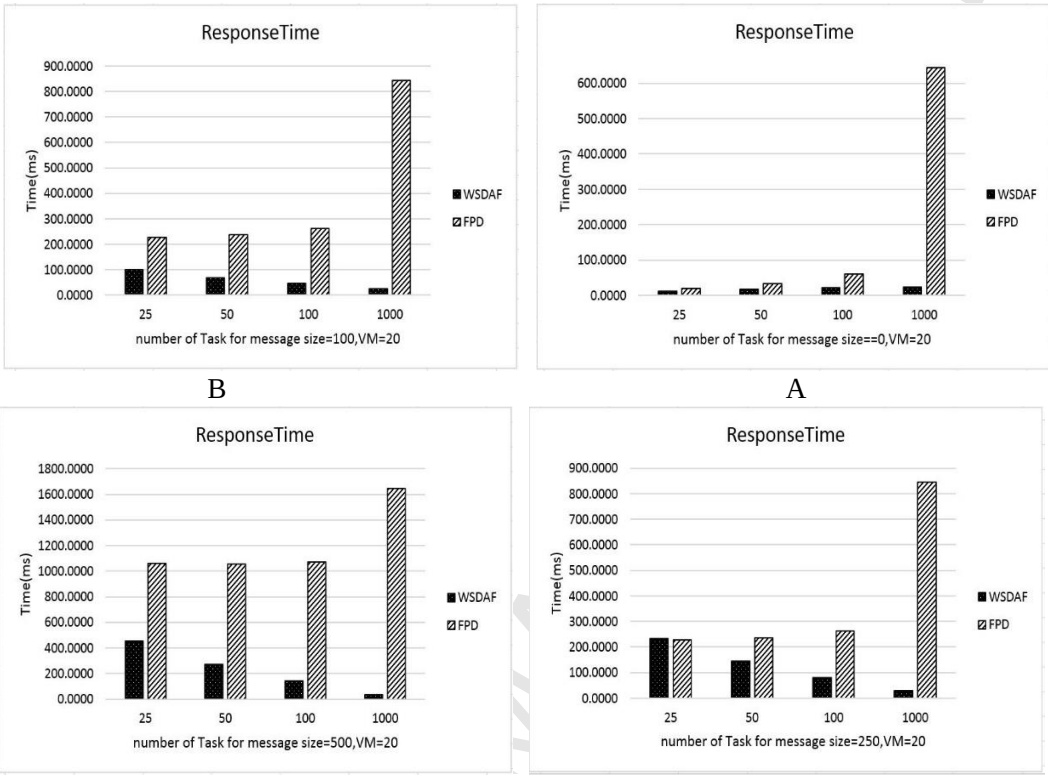
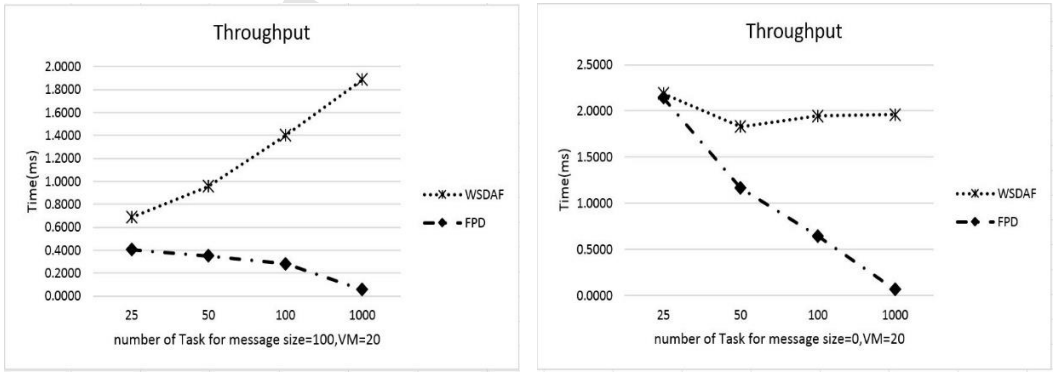
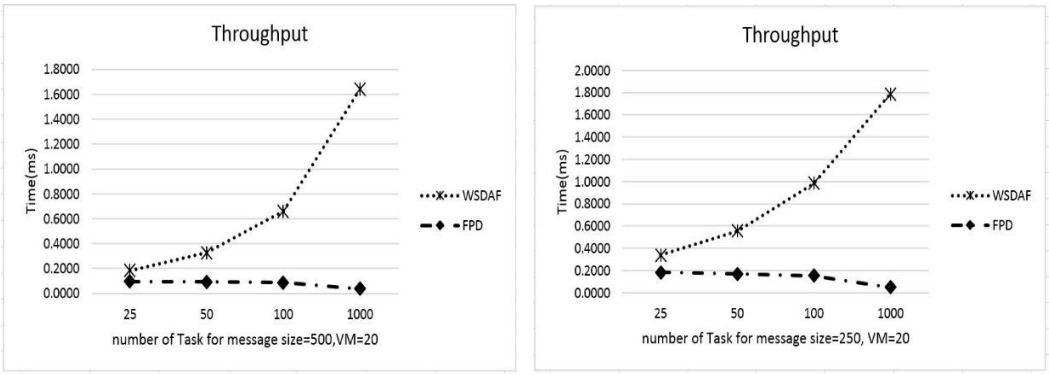


Fig 16: (A-D) the results of the mean response time from the experiment-2

The following Fig17 shows the results of Experiment 2 on the proposed framework and the FPD [4] algorithm with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Throughput. Based on Fig 17, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the FPD algorithm in terms of mean Throughput.





D C

Fig 17 :(A-D) the results of the mean Throughput from the Experiment-2

Experiment-2 examines the fragmentation phase with Configuration-2 and Montage (Mon), Epigenomics (EPI), Inspiral (INS), CyberShake (CYB), and Sipht (SIP) scientific workflows as datasets. Results of the improvement of the proposed framework are compared with those of the FPD [4] algorithm in Table 8. Results show the improved performance of the proposed framework compared to the FPD algorithm since the former controls fragment generation by considering the number of virtual machines constant and message size variable.

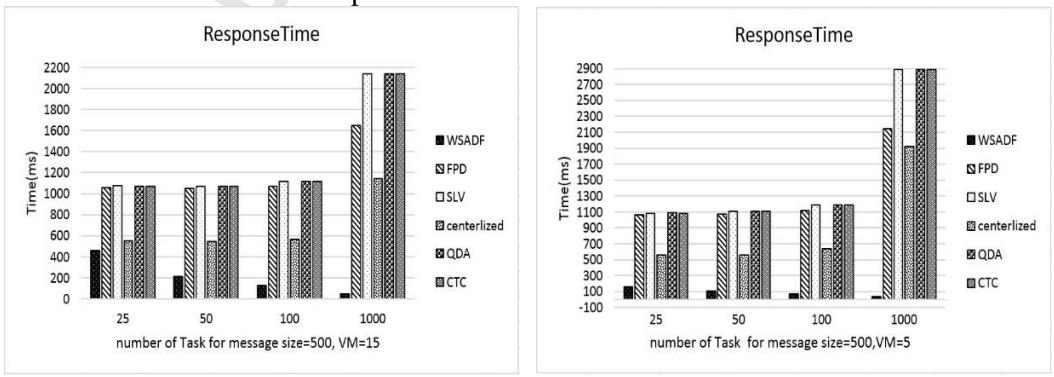
Table 8: Results of the improvement of WSADF framework than the FPD (Fragmentation-Configuration-2)

Fragmentation – Configuration 2										
	Response Time					Throughput				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	84.64	94.92	83.01	77.63	92.02	83.46	0.49	4.71	35.47	4.94

5-4- Evaluation of scheduling phase

5-4-1. Experiment-3: Scheduling using Configuration-1 (variable number of virtual machines; constant message size)

Scheduling phase with Configuration-1 reduces mean runtime and increases throughput, while message size is assumed constant and the number of virtual machines is assumed variable. The following Fig18 shows the results of Experiment 3 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean response time. Based on Fig 18, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean response time.



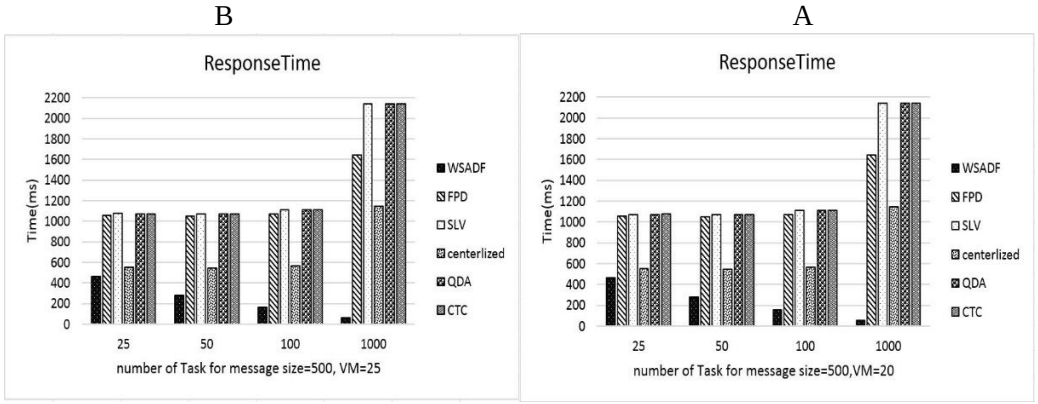


Fig18 :(A-D) the results of the mean response time from the Experiment-3

The following Fig19 shows the results of Experiment 3 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Throughput. Based on Fig 19, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean Throughput.

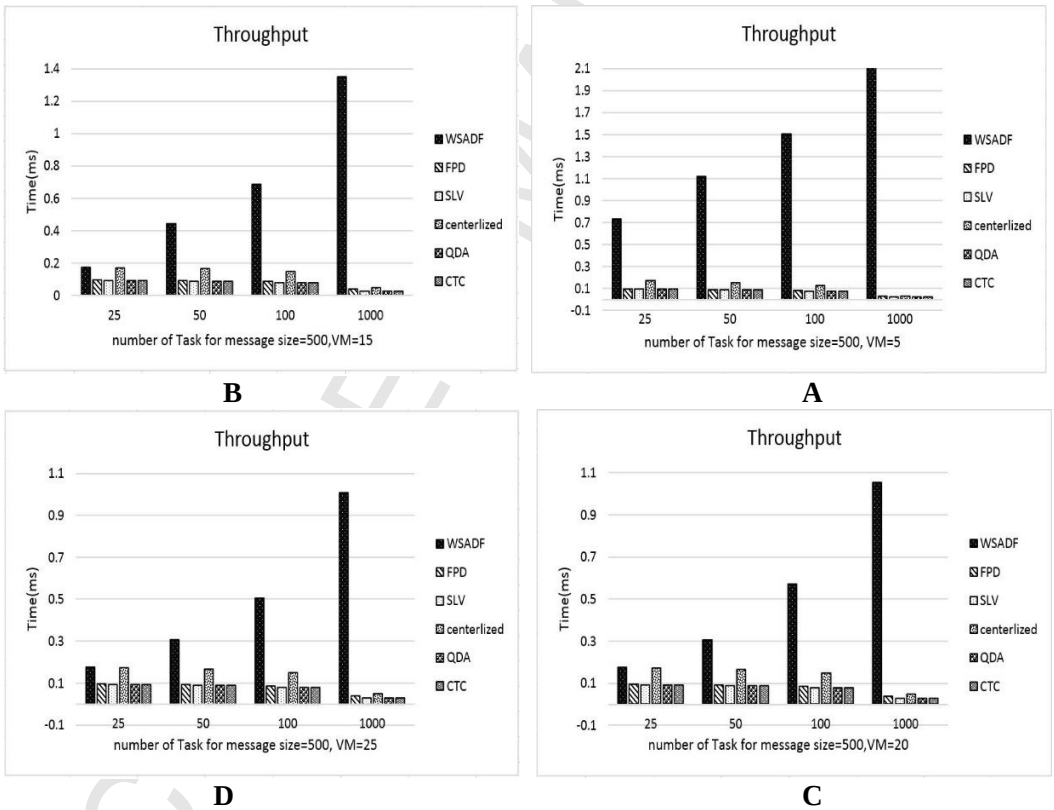


Fig19 :(A-D) the results of the mean Throughput from the Experiment-3

The following Fig20 shows the results of Experiment-3 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean used bandwidth cost (MB). Based on Fig 20, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean used bandwidth cost (MB).

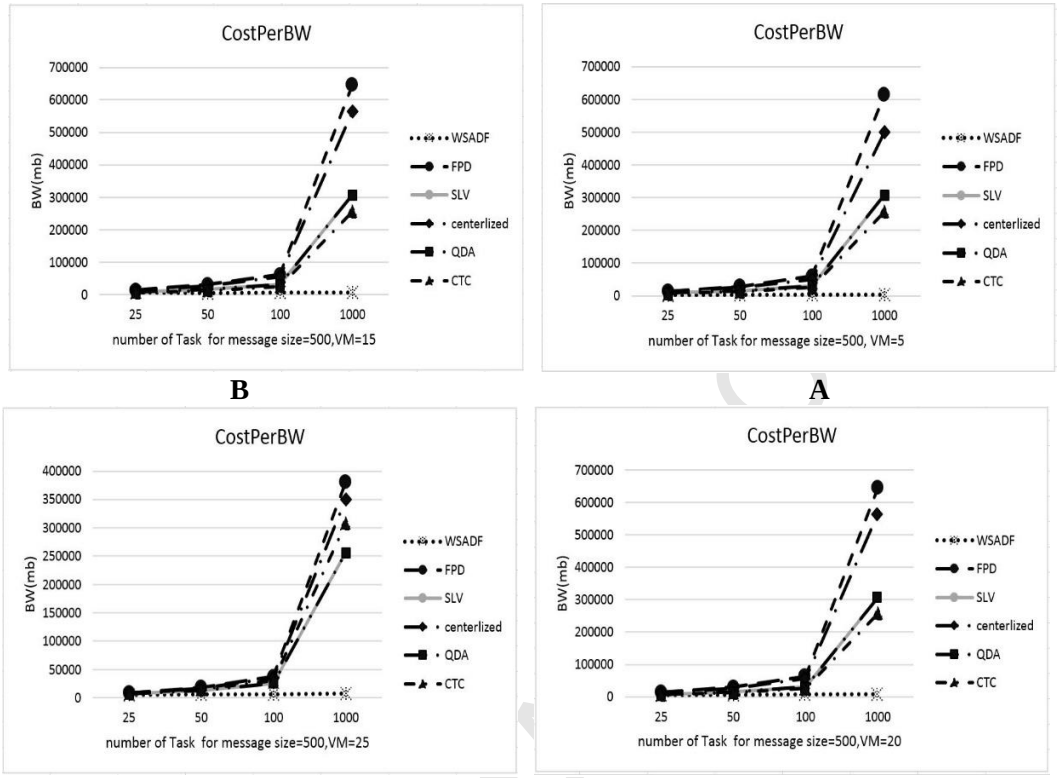
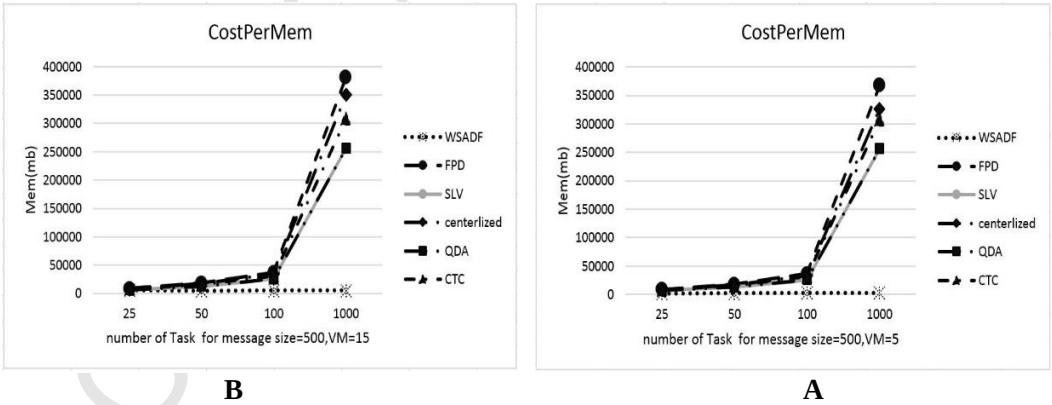
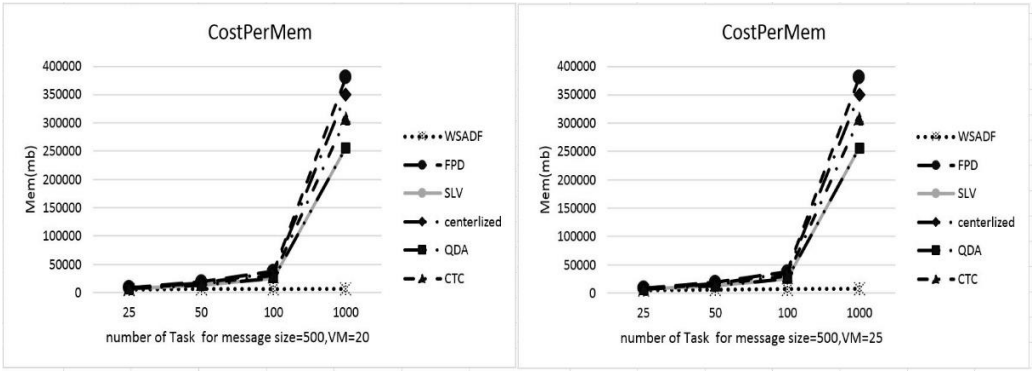


Fig20 :(A-D) the results of the mean used bandwidth cost (MB) from the Experiment-3

The following Fig21 shows the results of Experiment-3 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Memory used for any virtual machine (MB). Based on Fig 21, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean Memory used for any virtual machine (MB).





D **C**

Fig21 :(A-D) the results of the mean Memory used for any virtual machine (MB) from the experiment3

Experiment-3 examines the scheduling phase with Configuration-1 and Montage (Mon), Epigenomics (EPI), Inspiral (INS), CyberShake (CYB), and Sipht (SIP) scientific workflows as datasets. Results of the improvement of the proposed framework are compared with those of the other models in Tables 9 and 10. Compared to FPD [4], SLV [7], QDA [8], Centralized [31], and CTC [6] algorithms, the WSADF framework has improved performance by considering the number of virtual machines variable and message size constant, while controlling fragment generation and selecting a low-cost resource during runtime.

Table 9: Results of the improvement of the proposed framework than other Algorithm for the response time and throughput factors (Configuration-1)

Scheduling- Configuration 1- Table1										
	Response Time					Throughput				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	83.94	91.01	82.78	81.19	88.57	69.57	0.37	5.7	40.9	6.45
SLV	85.68	94.77	88.77	85.1	92.71	70.15	0.58	5.82	41.67	7.16
Centralized	72.83	91.35	82.04	70.43	87.7	64.17	0.52	5.99	38.7	6.63
QDA	85.69	98.22	93.62	85.47	94.04	70.15	0.59	5.81	41.65	7.14
CTC	85.68	73.26	88.95	63.32	93.05	70.15	0.25	5.81	41.69	7.18

Table 10: Results of the improvement of the proposed framework than other Algorithm for the cost factors (Configuration-1)

Scheduling- Configuration 1- Table2										
	CostPerBW					CostPerRam				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	96.91	98.63	96.26	96.69	98.08	95.48	98.08	94.74	95.31	97.27
SLV	93.6	97.21	92.26	93.01	95.92	93.34	97.21	92.26	93.01	95.92
Centralized	96.44	98.33	95.59	95.94	97.77	95.08	97.85	94.19	94.69	97.01
QDA	93.6	97.21	92.26	87.58	95.92	93.34	97.21	92.26	93.01	95.92
CTC	92.32	96.65	90.71	91.61	95.11	94.45	97.67	93.55	71.84	96.6

5-4-2. Experiment-4: Scheduling using Configuration-2 (constant number of virtual machines; variable message size)

Scheduling phase with Configuration-2 reduces mean runtime and increases throughput, while message size is assumed variable and the number of virtual machines is assumed constant. The following Figs22 show the results of Experiment 4 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean response time. Based on Fig 22, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean response time.

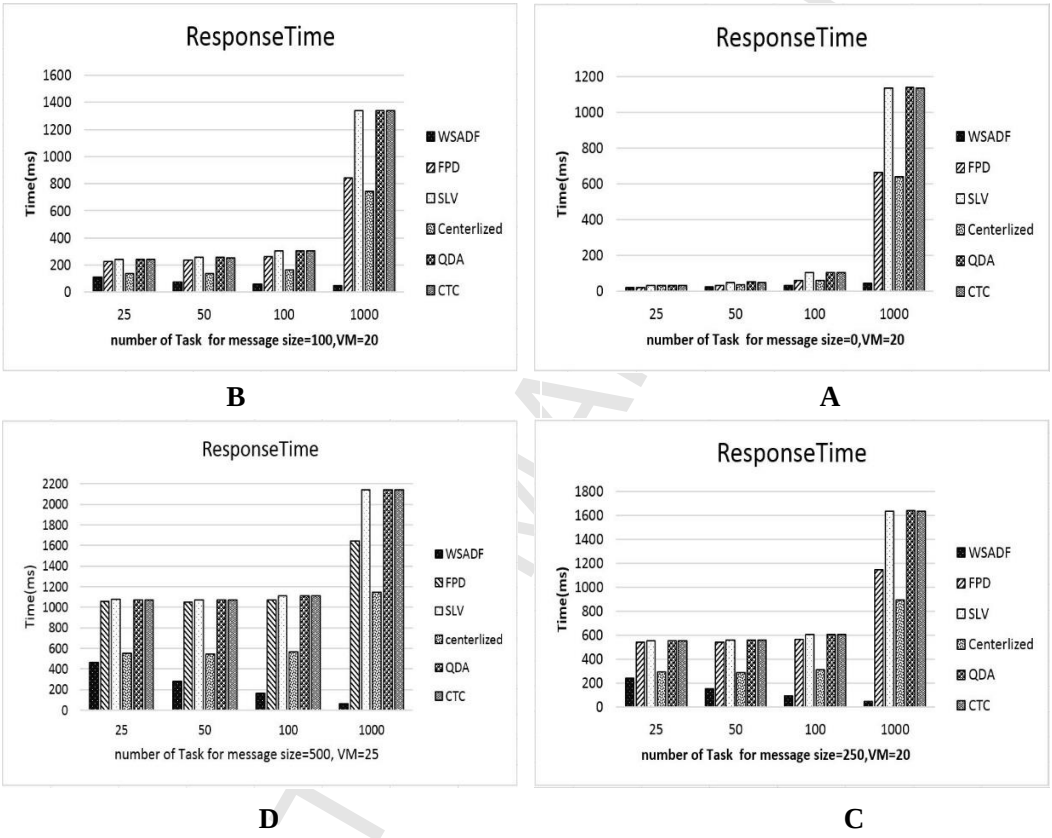


Fig 22 :(A-D) the results of the mean response time from the Experiment-4

The following Figs23 show the results of Experiment 4 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Throughput. Based on Fig 23, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean Throughput.

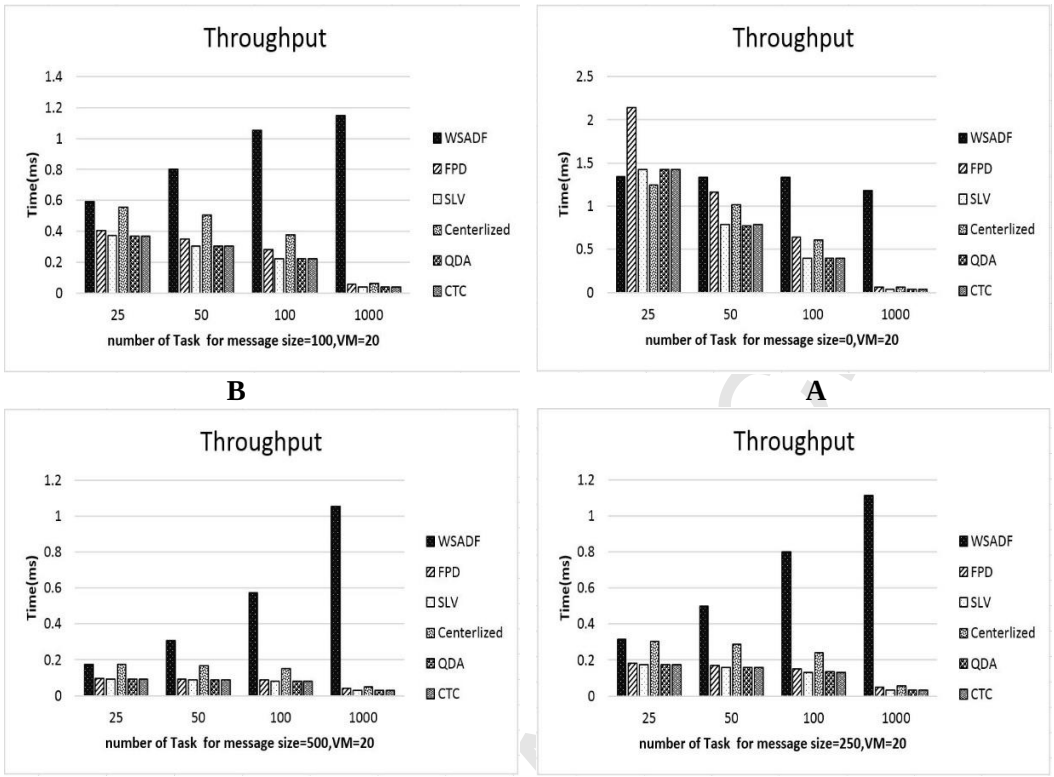
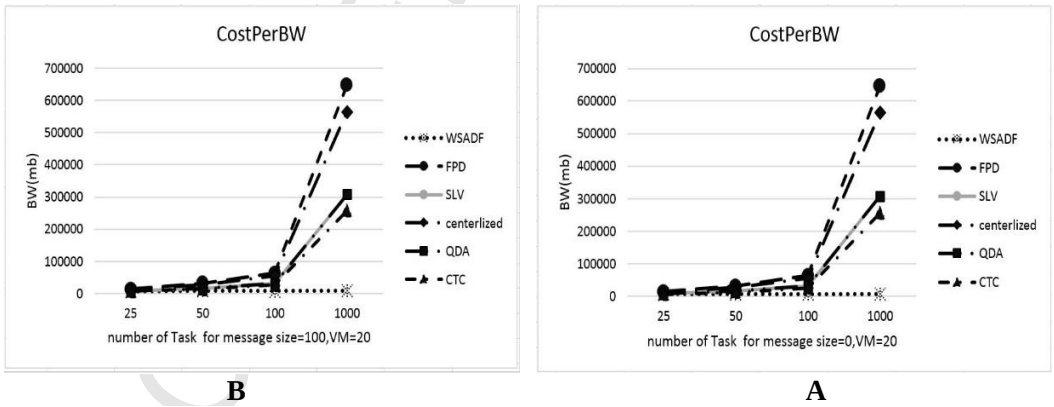
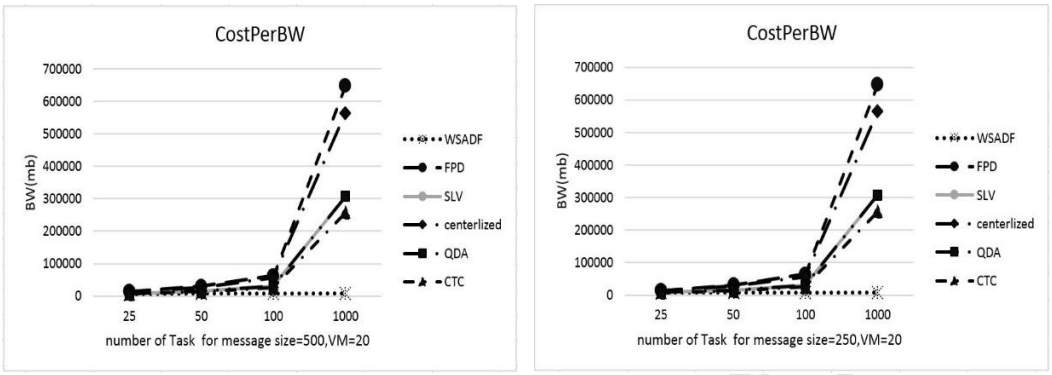


Fig23 :(A-D) the results of the mean Throughput from the Experiment-4

The following Figs24 show the results of Experiment-4 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean used bandwidth cost (MB). Based on Fig 24, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean used bandwidth cost (MB).

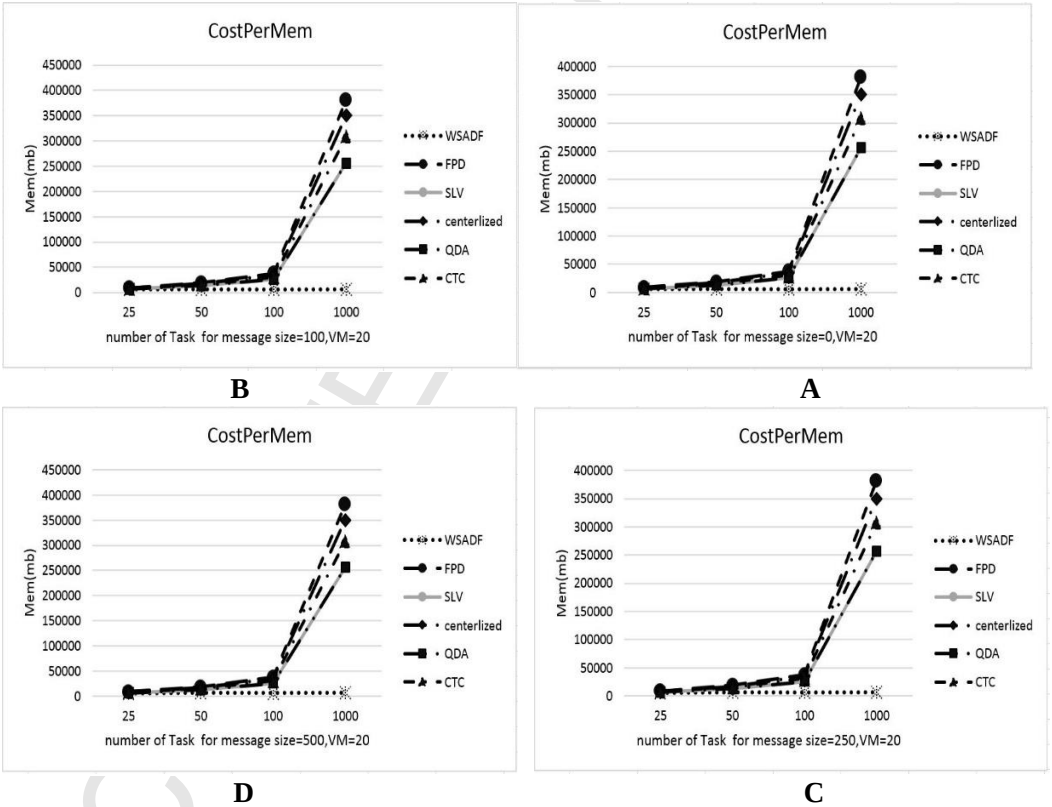




D **C**

Fig24 :(A-D) the results of the mean used bandwidth cost (MB) from the Experiment-4

The following Figs25 show the results of Experiment-4 on the proposed framework and the FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms with Montage dataset in four groups of task numbers (25, 50, 100, and 1000) for mean Memory used for any virtual machine (MB). Based on Fig 25, by increasing the number of tasks in the workflow, the WSADF framework has much better results than the SLV, FPD, CTC, Centralized, and QDA algorithms in terms of mean Memory used for any virtual machine (MB).



B **A**

D **C**

Fig25 :(A-D) the results of the mean Memory used for any virtual machine (MB) from the Experiment-4

Experiment-4 examines the scheduling phase with Configuration-2 and Montage (Mon), Epigenomics (EPI), Inspiral (INS), CyberShake (CYB), and Sipht (SIP) scientific workflows as datasets. Results of the improvement of the proposed framework are compared with those of the other models in Tables 11 and 12. Compared to FPD [4], SLV [7], Centralized [31], CTC [6], and QDA [8] algorithms, the WSADF framework has improved performance by considering the number of virtual machines constant and

message size variable, while controlling fragment generation and selecting a low-cost resource during runtime.

Table 11: Results of the improvement of the proposed framework than other Algorithm for the response time and throughput factors (Configuration-2)

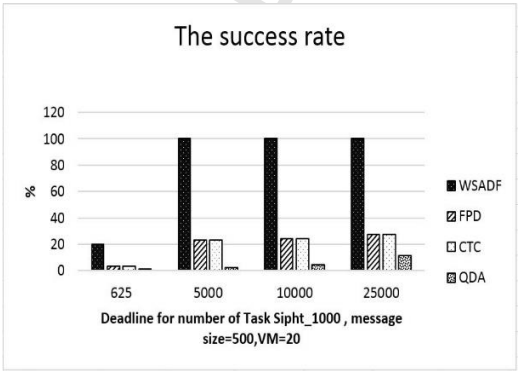
Scheduling- Configuration 2- Table1										
	Response Time					Throughput				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	81.47	90.72	81.16	75.81	87.88	47.82	-0.04	3.65	31.27	1.92
SLV	85.72	94.6	88.49	83.15	88.24	57.86	0.21	3.78	33.73	3.19
Centralized	73.62	91.17	81.5	66.31	87.49	48.56	0.15	4.15	30.17	2.91
QDA	85.74	98.19	93.71	83.76	90.39	57.95	0.22	3.93	33.84	3.09
CTC	85.72	96.32	88.84	83.06	93.21	57.84	0.2	4.09	34.04	3.16

Table 12: Results of the improvement of the proposed framework than other Algorithm for the cost factors (Configuration-2)

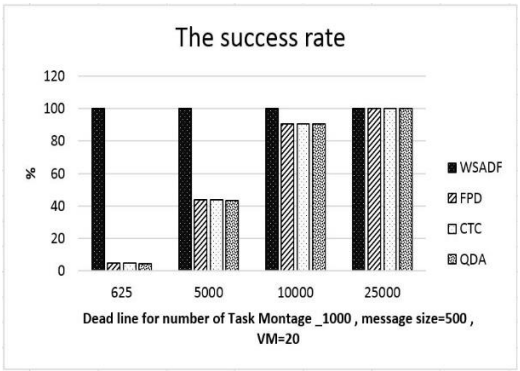
Scheduling- Configuration 2- Table2										
	CostPerBW					CostPerRam				
WF→	Mon	EPI	SIP	CYB	INS	Mon	EPI	SIP	CYB	INS
FPD	96.1	98.62	95.56	95.87	97.68	94.49	98.05	93.67	94.17	96.71
SLV	91.86	97.1	90.48	92.89	95.1	91.86	97.1	90.48	91.22	95.1
Centralized	95.53	98.3	94.63	94.94	97.38	94.02	97.79	92.9	93.37	96.45
QDA	91.86	97.08	90.48	91.22	95.1	91.86	97.1	90.48	91.22	95.1
CTC	90.23	96.52	88.58	55.92	94.12	93.21	97.58	92.07	92.68	95.92

5-4-3. Experiment-5: Scheduling using Configuration-3 (constant number of virtual machines; constant message size; variable deadline)

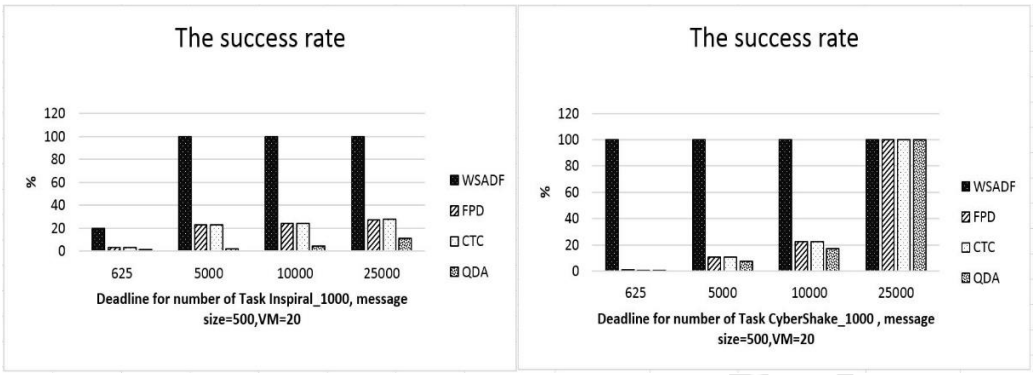
This experiment investigates the Montage, Sipht, Inspiral, and CyberShake scientific workflows as datasets considering Configuration-3 in the deadlines of 625, 5000, 10000, and 25000 ms. Results are presented in Fig 26, indicating the percentage of success of the WSADF framework in different deadlines compared with the baseline studies.



B



A



D **C**

Fig 26: (A-D) results of the configuration3

6- Conclusions and Future Studies

Workflow fragmentation and scheduling are significant problems in workflow management. Based on the definition by Wei Tan et al., fragmentation is a partition of workflow model. Models proposed for fragmentation have disadvantages such as increased number of generated fragments after fragmentation, which increases communication messages, time delay, and mean response time, thereby reducing throughput. Other disadvantages include static fragment generation which decreases flexibility and efficiency. Thus, the present study proposed a model for fragmentation which dynamically fragmented scientific workflows, considering runtime conditions. Moreover, it resolved the noted problems by controlling fragment generation. In WSADF framework, the number of tasks in each fragment was calculated based on the number of virtual machines. Fragments were generated during the execution, reducing communication messages among the fragments. In this study, WSADF framework was compared with the FPD algorithm in the fragmentation phase, and with FPD, CTC, Centralized, SLV, and QDA algorithms in the scheduling phase. According to the results of the experiments, response time and throughput were improved compared to the baseline studies. As the result of decreasing the number of generated fragments compared to the baseline studies, the number of communication messages among the fragments as well as delay time was reduced in this study, thereby decreasing response time and enhancing throughput. Furthermore, the results of the experiments for bandwidth usage cost and memory cost revealed the improved performance of the proposed framework compared to the baseline studies because the former controlled the number of generated fragments and selected appropriate virtual machines with less cost during runtime. Experiments were conducted in three Configuration and in both phases of fragmentor and scheduler. Results were improved compared to the baseline studies. For instance, compared to Montage workflow in Configuration-1 and the fragmentor phase, it showed 84.75% improvement in mean response time and 87.68% improvement in throughput. In Configuration-2 and the fragmentor phase, it showed 84.64% improvement in mean response time and 83.46% improvement in throughput. In Configuration-1 and the scheduler phase, it demonstrated 83.94%, 69.56%, and 96.91% improvement in mean response time, throughput, and bandwidth usage cost, respectively. In Configuration-2 and the scheduler phase, it demonstrated 94.49%, 47.82%, and 96.1% improvement in mean response time, throughput, and bandwidth usage cost, respectively. In Configuration-3 the results of experiments on datasets with variable deadline (625, 5000, 20,000 and 25,000) indicate that the success rate of the proposed framework was 100% than the base models.

Based on the wide range of scientific workflows, numerous problems can be studied. Limiting factors can be used to generate fragments from a workflow. In this way, there should be methods to balance different aspects of workflow execution such as scalability, distribution, bandwidth usage, budget etc. Furthermore, considering various and unequal resources is a major problem in workflow fragmentation and scheduling.

7- Resources

[1] F. Safi Esfahani, M. Masrah Azrifah Azmi, Md. Nasir Sulaiman, N. Izura Udzir, Adaptable decentralized service oriented, The Journal of Systems and Software 84 (10) (2011) 1591-1617.
[2] E. F. Duipmans, L F Pires, L.O.B. da Silva Santos. A Transformation-Based Approach to Business Process Management in the Cloud, Grid Computing (2013) 217–228.
[3] F. Wu, Q. Wu, Y. Tan, Workflow scheduling in cloud: a survey, Springer Science Business Media New York 71 (9) (2015) 3373-3418.
[4] G. LI, V. Muthusamy, H. Jacobsen, A Distributed Service Oriented Architecture for Business Process Execution, In Proc. of ACM Transactions on the Web 4 (1) (TWEB) (2010) 2.
[5] R. Khorsand, F. Safi Esfahani, N. Nematbakhsh, M Mohsenzade, ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering runtime circumstances in cloud computing environments, The Journal of Supercomputing 73 (6) (2017) 2430-2455.
[6] L. Ke, J. Hai, C. Jinjun, L. Xiao, Y. Dong, Y. Yun, A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on a Cloud Computing Platform, International Journal of High Performance Computing Applications 24 (4) (2010) 445-456.
[7] V. Muthusamy, H. Jacobsen, T. Chau, A. Chan, P. Coulthard, SLA-Driven Business Process Management in SOA, Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research (2009) 86-100.
[8] H Li, S Ge, Lu Zhang, A QoS-based Scheduling Algorithm for Instance-intensive Workflows in Cloud Environment, The 26th Chinese Control and Decision Conference (2014) 4094-4099.
[Dataset] [9] G. Juve, A. Chervenak, E. Deelman, SH. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2013) 682-692.
[10] W. Chen, E. Deelman, WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments, E-Science (e-Science), 2012 IEEE 8th International Conference on (2012) 1-8.
[11] E. Donyadari, F Safi Esfahani, N. Nourafza, Scientific Workflow Scheduling Based on Deadline Constraints in Cloud Environment, International Journal of Mechatronics Electrical and Computer Technology (IJMEC) 5 (16) (2015).
[12] R. Khorsand, F. Safi Esfahani, N. Nematbakhsh, M Mohsenzade, Taxonomy of Workflow Partitioning Problems and Methods in Distributed Environments, The Journal of Supercomputing 132 (2017) 253-271
[13] M. Naghibzadeh, Modeling and scheduling hybrid workflows of tasks and task interaction graphs on the cloud, Future Generation Computer Systems 65 (2016) 33-45.
[14] V Arabnejad, K Bubendorfer, B Ng, Scheduling Deadline Constrained Scientific Workflows on Dynamically Provisioned Cloud Resources, Future Generation Computer Systems 75 (2017)348-364.
[15] M. Atkinson, S. Gesing, J. Montagnat, I. Taylor, Scientific workflows: Past, present and future, Future Generation Computer Systems (2017)216 - 227.
[16] F. Safi Esfahani, M. Masrah Azrifah Azmi, Md. Nasir Sulaiman, N. Izura Udzir, SLA-Driven Business Process Distribution, Information Process and Knowledge Management 2009 EKNOW'09 International Conference on (2009) 14-21.
[17] F. Safi Esfahani, M. Masrah Azrifah Azmi, Md. Nasir Sulaiman, N. Izura Udzir, Using Process Mining to Business Process Distribution, Proceedings of the 2009 ACM symposium on Applied Computing (2009) 2140-2145.
[18] F. Safi Esfahani, M. Masrah Azrifah Azmi, Md. Nasir Sulaiman, N. Izura Udzir, Run-time adaptable business process decentralization, The Third International Conference on Information, Process, and Knowledge Management (2011) 76-82

[19] S. Abrishami, M. Naghibzadeh, D. H. J. Epema, Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds, Future Generation Computer Systems 29 (1) (2013) 158-169.
[20] Dr. D.I. George Amalarethinam1, P. Muthulakshmi, An Overview of the Scheduling Policies and Algorithms in Grid Computing, International Journal of Research and Reviews in Computer Science (IJRRCS) 2 (2011)280-294.
[21] V. Guth, K. Lenz, A. Oberweis, Distributed Workflow Execution Based On Fragmentation Of Petri Nets, Journal of Intelligent Information Systems 10 (2) (1998) 159–184**.**
[22] S. Torabi**,** F. Safi Esfahani, A Dynamic Task Scheduling Framework based on Chicken Swarm and Improved Raven Roosting Optimization methods in Cloud Computing, Springer Journal of Supercomputing (2018)1-46.
[23] N. Alaei, F. Safi Esfahani, RePro-Active: A Reactive-Proactive Scheduling Method Based on Pre-simulation in Cloud Computing, Springer Journal of Supercomputing 74 (2) (2018) 801-829.
[24] F. Motavaselalhagh, F. Safi Esfahani, H. R. Arabnia, Knowledge-based adaptable scheduler for SaaS providers in cloud computing, Springer Journal of Human-centric Computing and Information Sciences 5 (1) (2015) 16.
[25] P. Haratian, F. Safi Esfahani, L. Salimian, A. Nabiollahi, An adaptive and fuzzy resource management approach in cloud computing, E-Science (e-Science), IEEE Transactions on Cloud Computing (2017).
[26] M. Mozakka, F. Safi Esfahani, M. H. Nadimi, Survey on adaptive job schedulers in mapreduce, Journal of Theoretical & Applied Information Technology 66 (3) (2014).
[27] L. Salimian, F. Safi Esfahani, M. H. Nadimi, An adaptive fuzzy threshold-based approach for energy and performance efficient consolidation of virtual machines, Computing 10 (10) (2016) 641-660.
[28] L. Salimian, F. Safi Esfahani, Survey of energy efficient data centers in cloud computing, Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (2013) 369-374.
[29] W. Tan. Y. Fan, Model Fragmentation for Distributed Workflow Execution: A Petri Net Approach, Springer-Verlag Berlin Heidelberg (2005) 207–214.
[30] W. Tan, Y. Fan, Dynamic workflow model fragmentation for distributed execution, Comput. Ind 58 (5) (2007) 381-391.
[31] P. Muth, D. Wodtke, J. Weissenfels, AK. Dittrich,V. Gerhard, From Centralized Workflow Specification to Distributed Workflow Execution, Journal of Intelligent Information Systems 10 (2) **(**1998)159–184.
[32] K. Almi'Ani, Y C. LEE, Partitioning-Based Workflow Scheduling in Clouds, Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on (2016) 645-652.
[33] SH. Sun, Q. Zeng, H. Wang, Process-mining-based workflow model fragmentation for distributed execution, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 41 (2) (2011) 294-310
[34] C. Faisstnauer, D. Schmalstieg, and W. Purgathofer, Priority round-robin scheduling for very large virtual environments, Virtual Reality Conference (2000) 135-142.
[35] A. Oprescu and T. Kielmann, Bag-of-tasks scheduling under budget constraints, Cloud Computing Technology and Science (CloudCom), IEEE Second International Conference (2010) 351-359.
[36] S. Bansal, B. Kothari, and C. Hota, Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm, International Journal of Computer Science 8 (2011) 472-477.
[37] S. Abrishami, M. Naghibzadeh, Deadline-constrained workflow scheduling in software as a service Cloud, Scientia Iranica 19 (2012) 680-689.

[38] H. A. Abba, N. B. Zakaria, A. J. Pal, and K. Naono, Performance Comparison of Some Hybrid Deadline Based Scheduling Algorithms for Computational Grid, International Conference on Advances in Information Technology (2012) 19-30.
[39] R. N. Calheiros1, R. Ranjan2, A. Beloglazov1, C´esar A. F. De Rose3and R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, The Software: Practice and experience 41 (1) (2011) 23–50.

Zahra Momenzadeh has received her master degree in software engineering from Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran. Her current research interests include cloud computing, scheduling and fragmentation algorithms, and Workflow Scientific.

Faramarz Safi-Esfahani received his Ph.D. in Intelligent Computing from University of Putra Malaysia in 2011. He is currently on faculty at Department of Computer Engineering, Islamic Azad University, Najafabad Branch, Iran. His research interests are intelligent computing, Big Data and Cloud Computing, Autonomic Computing, and Bio-inspired Computing

This paper represents a workflow execution approach in cloud computing and includes:

1) WSADF framework with two phases of workflow fragmentation and scheduling.
2) The fragmentation phase generates appropriate fragments considering runtime conditions.
3) The scheduling phase schedule fragments in order to reduce runtime costs.
4) CyberShake, Sipht, Montage, Epigenomics, and Inspitral are used for the evaluations.