BDAP: A Big Data Placement Strategy for Cloud-Based Scientific Workflows

Mahdi Ebrahimi, Aravind Mohan, Andrey Kashlev, and Shiyong Lu
Wayne State University
Detroit, U.S.A.
{mebrahimi, amohan, andrey.kashlev, shiyong}@wayne.edu

Abstract— **In this new era of Big Data, there is a growing need to enable scientific workflows to perform computations at a scale far exceeding a single workstation's capabilities. When running such data intensive workflows in the cloud distributed across several physical locations, the execution time and the resource utilization efficiency highly depends on the initial placement and distribution of the input datasets across these multiple virtual machines in the Cloud. In this paper, we propose BDAP (Big DAta Placement strategy), a strategy that improves workflow performance by minimizing data movement across multiple virtual machines. In this work, we 1) formalize the data placement problem in scientific workflows, 2) propose a data placement algorithm that considers both initial input dataset and intermediate datasets obtained during workflow run, and 3) perform extensive experiments in the distributed environment to verify that our proposed strategy provides an effective data placement solution to distribute and place big datasets at the appropriate virtual machines in the Cloud within reasonable time.**

Keywords—Big Data, Data Placement, Cloud Computing, Scientific Workflow, Evolutionary Algorithm

I. Introduction

Workflows have been extensively employed in various scientific areas such as bioinformatics, physics, astronomy, ecology, and earthquake science [10]. They are usually modeled as directed acyclic graphs (DAGs) such that workflow tasks are represented as graph vertices and the data flows among tasks are represented by graph edges. The direction of edges shows data flows among tasks. A scientific workflow management system (SWFMS) is a system to design and execute scientific workflows (SWF).

Scientific workflows are potentially very large and comprise hundreds or thousands of complex tasks and big datasets [3, 6]. Moving huge datasets across workflow tasks increases the execution time of scientific workflows. To improve throughput and performance, this type of application can greatly benefit from distributed high performance computing (HPC) infrastructures such as Cloud computing.

Could Computing has been studied as the next generation architecture of IT enterprise by providing cost-effective, scalable, on-demand, elastic provisioning and distributed computing infrastructure [9]. To execute a scientific workflow in the Cloud, scientific workflow tasks and datasets should be partitioned, distributed, and assigned to various execution sites in the Cloud. The advantages of using Cloud computing for scientific workflows are summarized as follows [7]: 1) providing large amount of storage space and computing resources; 2) improving resource utilization by allocating resources on as-needed basis; 3) providing a much larger room for the trade-off between performance and cost.

Since scientific applications become more and more data intensive, data management is sometimes more critical than other resource management in Cloud computing. Scientific datasets are often in terabytes or even petabytes in size, creating the need for dedicated virtual machines just for data storage purposes [14]. Therefore, data management needs to utilize an effective data placement strategy in order to maximize data locality and minimize data migration between virtual machines in the Cloud. Data placement should be applied such that once workflow tasks are partitioned and assigned to a virtual machine in the Cloud, their required datasets are already stored in the same virtual machine. In large workflows, it is practically impossible to store all of the required datasets of tasks in one virtual machine due to the storage capacity limitation of virtual machines and so data movement is necessary to execute scientific workflows. The main goal of data placement is to minimize the total data movement between workflow tasks because "moving computation to data is often cheaper than moving data to computation" [1].

Scientific workflow typically models and analyzes complex scientific research experiments along with huge volumes of datasets. These large datasets are physically distributed and placed on different sources by users over the Internet. On the other hand, new advancements in IT technology bring the capability of collecting and accessing these huge amounts of information and datasets not only by human beings but also by computers. Big datasets are difficult to manage by traditional data management tools and strategies. Therefore big data technology is becoming the main focus in scientific computing research such as the scientific workflow domain recently [2]. Big Data can be defined by 5 characteristics, called the five V's: Volume, Value, Velocity, Variety, and Veracity [15].

978-1-4799-8128-1/15 $31.00 © 2015 IEEE
DOI 10.1109/BigDataService.2015.70

The most important feature of Big Data is volume, which characterizes the size of datasets such as records, tables or files. Value reflects the usefulness of a dataset. Analyzing and processing of collected big datasets for the purpose of scientific discovery is the main motivation for researchers that use scientific workflow technology. Velocity applies to the enormous volume of dataset that come in/out with high speed from different sources. Variety is about vast different dataset formats and large number of diverse dataset sources to integrate. And finally, veracity is related to datasets consistency/certainty and trustworthiness that can applied in various stages of data management like dataset collecting and processing stages. Our study is about the volume aspect of Big Data and how it operates to partition, distribute, and place huge size of different datasets into virtual machines in the Cloud such that the most interdependent datasets are clustered and placed to the same virtual machine.

In this paper, we propose BDAP, an evolutionary algorithm (EA) which is a generic population-based metaheuristic optimization strategy for data placement in distributed scientific workflows. The main goal is to minimize the dataset movement between virtual machines during the execution of a workflow under the constraint of virtual machine storage capacity.

Example I. Let's consider an example to show how a scientific workflow can be executed in a Cloud computing environment. Figure I illustrates a sample scientific workflow with five tasks, five original datasets and five generated intermediate datasets [14]. Figure II shows an instance of its virtual machines configuration in the Cloud. In this example, tasks t_1 and t_2 as well as datasets, d_1 and d_3 were assigned to virtual machine 1, VM_1. Similarly, tasks t_3 and t_4 were assigned to VM_2 as well as datasets, d_2 and d_4. Once we execute the workflow, tasks t_2 needs transferring dataset d_2 from VM_2 to VM_1 to complete its process. However there is no need to move any other original datasets from other virtual machine to VM_2 to run task t_3 because all of its required original datasets, d_2 and d_4 are already placed in VM_2. Furthermore, t_3 only required transferring the output of task t_1, d_1 from VM_1 to VM_2 in the run-time stage. We propose a data placement strategy for scientific workflows in this paper. We explain our proposed strategy for data placement in section 3 in detail.

To come up with a data placement of scientific workflows in the Cloud computing infrastructure, our proposed strategy, BDAP, clusters the most interdependent datasets (workflow tasks) together and store them possibly in the same virtual machine in the Cloud. BDAP applies the one algorithm for placement of original and generated intermediate datasets.

A random set of data placement schemes are generated in the first step. In the next step, BDAP computes and compares the generated schemes by applying a defined heuristic function and return the best scheme. The heuristic function is based on the data interdependency within and between the virtual machines in the Cloud. The best scheme is the one which maximizes the data interdependency within each virtual machine and minimizes the data interdependency between virtual machines. After placing the datasets, BDAP assigns and schedules workflow tasks into corresponding virtual machines.

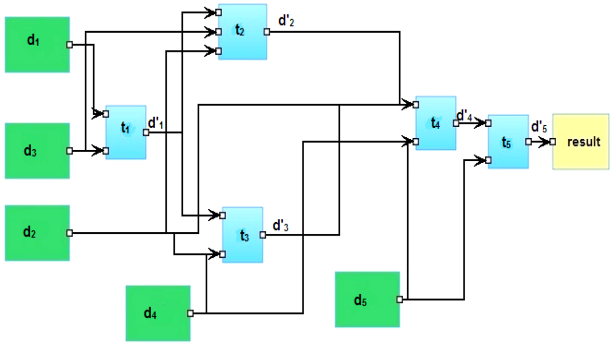

Figure I. A scientific workflow with five tasks $\{t_1, t_2, t_3, t_4, t_5\}$ and five datasets $\{d_1, d_2, d_3, d_4, d_5\}$. The output of task ti is denoted by d'i. The input datasets of task t_1 are $\{d_1, d_3\}$, t_2 are $\{d'_1, d_2, d_3\}$, t_3 are $\{d'_1, d_2, d_4\}$, t_4 are $\{d'2, d'3, d_2, d_4, d_5\}$ and $t_5$are $\{d'_4, d_5\}$.

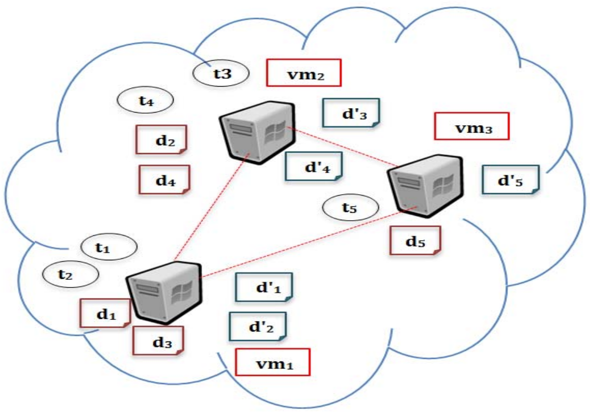

Figure II. A virtual machine configuration in the Cloud with three virtual machines for workflow of Example I. Datasets $\{d_1, d_3\}$ and tasks $\{t_1, t_2\}$ were placed and assigned in VM_1. Datasets $\{d_2, d_4\}$ and tasks $\{t_3, t_4\}$ were placed and assigned in VM_2. Similarly, dataset $\{d_5\}$ and tasks $\{t_5\}$ was placed and assigned in VM_3.

Please note although BDAP can be used for task assignment, the workflow scheduling is out of the scope of this paper. BDAP does not apply any specific strategy for the order (either sequential or parallel) execution of workflow tasks. BDAP can be used by any current workflow scheduling algorithms to improve the workflow throughput. In this paper, we simply execute workflow tasks in a sequential order to evaluate BDAP.

The rest of the paper is organized as follows: in Section 2 we define and formalize our system model. In Section 3 we explain our data placement strategy in detail. Then in Section 4, the experimental results are shown and discussed. Section 5 presents implementation of a scientific workflow in the FutureGrid. Section 6 presents related work. Finally, conclusion and future work are presented in Section 7.

II. FORMALIZING WORKFLOW DATA PLACEMENT

To model the Cloud computing environment, we consider I distributed virtual machines in the Cloud as the execution sites to execute workflows. Each virtual machine can be provided by different Cloud Computing Providers (CCP) such as Amazon EC2, Google App Engine, and Microsoft Azure. Although CCPs normally have their own data and computation placement strategy in order to store data and assign computation jobs to proper virtual machines, sometimes users (e.g., scientists) have concerns about their own datasets (e.g., data security or too large data or requirement for specific data processing utilities and equipment). Such users prefer to keep and store their data in a particular virtual machine and not allowed to move their data to the other virtual machines. This type of dataset is called *fixed-location datasets*.

For addressing these scientific user's concerns and managing fixed-location datasets, users need to have private execution sites or to be able to add their own local computation facilities as virtual machines. In that way, we need to apply a new data placement strategy to address the fixed-location datasets and minimize the total data movement across dedicated virtual machines in the Cloud.

To minimize data movement between virtual machines in the Cloud, we cluster the virtual machines such that the placed datasets have the highest data interdependency within each virtual machine as well as the lowest data interdependency between virtual machines. In the rest of this section, we model our data placement solution in detail. Table I summarizes all the used symbols and notations in this paper.

To execute a scientific workflow in Clouds, we need to model Clouds first. A Cloud computing environment is modeled as follows:

Definition 2.1 (Cloud Computing Environment C). A Cloud computing environment C is a 3-tuple $\mathrm{C} = (\mathrm{VM}, \mathrm{SC}, \mathrm{DTR})$, where

- VM is a set of virtual machine in the Cloud $\mathrm{vm_i}$ $(\mathrm{i} = 1, 2, \dots, \mathrm{I})$
- $\mathrm{SC}: \mathrm{VM} \rightarrow \mathrm{R}^+$ is a storage capacity function. SC $(\mathrm{vm_i})$, $\mathrm{vm_i} \in \mathrm{VM}$ gives the maximum available storage capacity of virtual machine $\mathrm{vm_i}$ in the Cloud computing environment C. It is measured in some pre-determined unit such as mega-bytes, giga-bytes or tera-bytes. R^+ is the set of positive real number.
- $\mathrm{DTR}: \mathrm{VM} \times \mathrm{VM} \rightarrow \mathrm{Q}_0^+$ is the data transfer rate function. $\mathrm{DTR}(\mathrm{vm_{i1}}, \mathrm{vm_{i2}})$, $\mathrm{vm_{i1}}$, $\mathrm{vm_{i2}} \in \mathrm{VM}$ gives the data transfer rate between two virtual machines $\mathrm{vm_{i1}}$ and $\mathrm{vm_{i2}}$. It is measured in some pre-determined unit such as mega-bytes, giga-bytes per second. Q_0^+ is the set of positive rational number.

For solving the complex scientific problems, scientists are able to create and run their own scientific workflows simultaneously. Each individual workflow contains a set of tasks that consume various datasets and may produce intermediate datasets as well. Those produced datasets will be sent to other tasks as their inputs by following the data flow logic. A scientific workflow is formalized as follows:

Definition 2.2 (Scientific Workflow W). A scientific workflow W can be modeled formally as a 6-tuple that consists of three sets and two functions as follows:

$$\mathrm{W} = (\mathrm{T}, \mathrm{D}, \mathrm{D}', \mathrm{S}, \mathrm{TS}, \mathrm{DS})$$

- T is the set of workflow tasks. Each individual task is denoted by $\mathrm{t_k}$, $\mathrm{T} = \{\mathrm{t_1}, \mathrm{t_2}, \mathrm{t_3}, \dots, \mathrm{t_K}\}$.
- D is the set of input datasets for workflow W. Each individual dataset is denoted by $\mathrm{d_j}$, $\mathrm{D} = \{\mathrm{d_1}, \mathrm{d_2}, \dots, \mathrm{d_J}\}$.
- D' is the set of output datasets for workflow W. The total number of output datasets is equal to the total number of workflow tasks as each workflow task, $\mathrm{t_k}$ generates one output dataset, $\mathrm{d_k}$ which can flow to the other tasks as the input dataset. Each individual output dataset is denoted by $\mathrm{d'_k}$, $\mathrm{D}' = \{\mathrm{d'_1}, \mathrm{d'_2}, \dots, \mathrm{d'_K}\}$.
- $\mathrm{S}: \mathrm{D} \cup \mathrm{D}' \rightarrow \mathrm{R}^+$ is the dataset size function.
- $\mathrm{S}\ (\mathrm{d_j})$, $\mathrm{d_j} \in \mathrm{D} \cup \mathrm{D}'$ returns the size of original or generated dataset $\mathrm{d_j}$. The size of a dataset is defined in some pre-determined unit such as mega-bytes, giga-bytes or tera-bytes. R^+ is the set of positive real number.
- $\mathrm{TS}: \mathrm{D} \cup \mathrm{D}' \rightarrow \mathrm{T}$ is the dataset-task function. $\mathrm{TS}(\mathrm{d_j})$, $\mathrm{d_j} \in \mathrm{D} \cup \mathrm{D}'$ returns the set of workflow tasks that consume $\mathrm{d_j}$ as their input.
- $\mathrm{DS}: \mathrm{T} \rightarrow \mathrm{D} \cup \mathrm{D}'$ is the task-dataset function. $\mathrm{DS}\ (\mathrm{t_k})$, $\mathrm{t_k} \in \mathrm{T}$ returns the set of datasets that are consumed by t_k as its input. The datasets can be either original or generated datasets.

To evaluate and compare BDAP with the others proposed algorithms Workflow Communication Cost is defined as follows:

Definition 2.3 (Workflow Communication Cost, WCC).

If dataset d_j is required to transfer from virtual machine vm_{i1} to vm_{i2} then the data movement cost of d_j is defined as

$$DMC\left(d_j, vm_{i1}, vm_{i2}\right) = \begin{cases} 0, & if\ i_1 = i_2 \\ \dfrac{S\left(d_j\right)}{DTR(vm_{i1}, vm_{i2})}, & if\ i_1 \neq i_2 \end{cases} \qquad (1)$$

Given a workflow W and Cloud computing C, workflow communication cost is equal to the total data movement cost for executing workflow W in C is defines as follows:

$$WCC(W, C) = \sum_{k=1}^{K} \sum_{\substack{d_j \in DS(t_k) \\ t_k \in W}} DMC\left(\, d_j, vm_{i1}, vm_{i2}\right) \qquad (2)$$

WCC gives the total data movement within executing the whole workflow in the Cloud C. In the remainder of this section, we define and model the problem and our solution. Our solution is based on the clustering technique. The three main concepts in clustering are objects which need to be clustered, clusters and a separation measure to compute the similarity among the objects. In this work, datasets are considered as the objects and virtual machines in the Cloud are considered as the clusters. The most important concept is defining a good separation measurement to cluster the most similar objects together to meet the objective goal.

TABLE I. SYMBOLS AND NOTATIONS SUMMARY

Notations	Description
$\boldsymbol{VM}$	The set of virtual machines
$\boldsymbol{vm_i}$	The ith virtual machine in VM
$\boldsymbol{SC(vm_i)}$	The storage capacity of virtual machine $\mathrm{vm_i}$
$\boldsymbol{D}$	The set of datasets
$\boldsymbol{D_{fixed}}$	The set of fixed datasets, $\mathrm{D_{fixed}} \subseteq \mathrm{D}$
$\boldsymbol{d_j}$	The jth dataset in D
$\boldsymbol{S(d_j)}$	The size of dataset $\mathrm{d_j}$
$\boldsymbol{T}$	The set of Tasks
$\boldsymbol{DS(t_k)}$	The set of datasets as the input of task $\mathrm{t_k}$
$\boldsymbol{TS(d_j)}$	The set of tasks which get dataset $\mathrm{d_j}$ as the input
$\boldsymbol{DTR(vm_{i1}, vm_{i2})}$	The data transfer rate between two virtual machines, $\mathrm{vm_{i1}}$and $\mathrm{vm_{i2}}$
$\boldsymbol{dp(d_{j1}, d_{j2})}$	The data interdependency between datasets $\mathrm{d_{j1}}$ and $\mathrm{d_{j2}}$
$\boldsymbol{DP}$	The data interdependency matrix of D
$\boldsymbol{\Psi}$	The J-element vector of datasets placement scheme which J is the number of workflow datasets.
$\boldsymbol{\Psi(d_j)}$	The virtual machine to which dataset d_j is assigned in the placement scheme Ψ
$\boldsymbol{P}$	The set of data placement schemes

The goal of this study is minimizing data movement among virtual machines. Therefore, we consider data interdependency as the separation measurement. For this, two datasets are interdependent and should be collocated in the same virtual machine if they are simultaneously needed as inputs by many tasks. The definition for the interdependency of a pair of datasets is as follows:

Definition 2.3 (Data Interdependency). We consider the number of common tasks that take a pair of datasets as input to define the data interdependency of the datasets. Data interdependency value is divided by the total number of workflow tasks in order to be normalized in the range of [0 1]. Formally, given two datasets d_{j1} and d_{j2}, the data interdependency is calculated by:

$$dp\left(d_{j1}, d_{j2}\right) = \frac{\left|TS(d_{j1}) \cap TS(d_{j2})\right|}{|T|} \tag{3}$$

For instance, if the set of tasks that consume $\mathrm{d_1}$ is $\mathrm{TS(d_1)} = \{\mathrm{t_1,t_2}\}$ and $\mathrm{d_2}$ is $\mathrm{TS(d_2)} = \{\mathrm{t_2,t_3,t_4}\}$ then the data interdependency between $\mathrm{d_1}$ and $\mathrm{d_2}$ is $\mathrm{dp(d_1, d_2)} = \frac{|\mathrm{TS(d_1)} \cap \mathrm{TS(d_2)}|}{|\mathrm{T}|} = \frac{|\{\mathrm{t_1,t_2}\} \cap \{\mathrm{t_2,t_3,t_4}\}|}{|\{\mathrm{t_1,t_2,t_3,t_4,t_5}\}|} = \frac{1}{5} = 0.20$. In this way, two datasets are interdependent once they have at least one common task consuming both of them. Two datasets have a higher interdependency when they are used by more common tasks and the greater the number of common tasks is, the higher is the data interdependency of datasets.

To maximize data locality, it is necessary to pre-cluster the datasets initially. In the first step, we calculate the data interdependency of all the workflow datasets and generate the data interdependency matrix (DM). In the interdependency matrix, rows and columns are the workflow datasets and the value of interdependency matrix is the data interdependency between two datasets. For instance, data interdependency matrix of workflow in Example1 is as follows:

$$\mathrm{DM} = \begin{array}{c} \mathrm{d_1} \\ \mathrm{d_2} \\ \mathrm{d_3} \\ \mathrm{d_4} \\ \mathrm{d_5} \\ \mathrm{d_1'} \\ \mathrm{d_2'} \\ \mathrm{d_3'} \\ \mathrm{d_4'} \end{array} \begin{pmatrix} 2 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 3 & 1 & 2 & 1 & 2 & 1 & 1 & 0 \\ 2 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 0 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

BDAP partitions and distributes the original datasets into all appropriate virtual machines in the Cloud. Then the related tasks will be assigned to the corresponding virtual machine so that their required datasets are stored there. In this way, the total amount of data movement between virtual machines is decreased and the overall workflow execution time will be reduced. Data placement scheme is defined to represent the place of each workflow dataset in a virtual machine. A data placement scheme is defined formally as follows:

Definition 2.4 (Data Placement Scheme $\boldsymbol{\Psi}$). Suppose there are I virtual machines and J datasets, a data placement scheme is represented by a J-element vector Ψ such that $\Psi(\mathrm{d_j})$ indicates the virtual machine to which $\mathrm{d_j}$ is placed. For example the data placement scheme of Example I is $\Psi = (1, 2, 1, 2, 3, 1, 1, 2, 2)$ and it means datasets $\mathrm{d_1}$, $\mathrm{d_3}$, $\mathrm{d_1^*}$ and $\mathrm{d_2^*}$ are placed in virtual machine $\mathrm{vm_1}$ ($\Psi(\mathrm{d_1}) = \Psi(\mathrm{d_3}) = \Psi(\mathrm{d_1'}) = \Psi(\mathrm{d_2'}) = \mathrm{vm_1}$), datasets $\mathrm{d_2}$, $\mathrm{d_4}$, $\mathrm{d_3'}$ and $\mathrm{d_4'}$ in virtual machine $\mathrm{vm_2}$ ($\Psi(\mathrm{d_2}) = \Psi(\mathrm{d_4}) = \Psi(\mathrm{d_3'}) = \Psi(\mathrm{d_4'}) = \mathrm{vm_2}$) and the dataset $\mathrm{d_5}$ in virtual machine $\mathrm{vm_3}$ ($\Psi(\mathrm{d_5}) = \mathrm{vm_3}$).

Definition 2.5 (Fixed-Location Datasets $\mathbf{D_{fixed}}$). Given the set of datasets, fixed-location datasets $\mathrm{D_{fixed}} \subseteq \mathrm{D}$ is a subset of D such that they have pre-determined allocations and cannot be moved. Formally suppose $\mathrm{D_{fixed}} = \{\mathrm{d_{j1}, d_{j2}, \dots, d_{jm}}\} \subseteq \mathrm{D}$ then

$$\Psi(\mathrm{d_{j1}}) = \mathrm{vm_{i1}}, \Psi(\mathrm{d_{j2}}) = \mathrm{vm_{i2}}, \dots \text{and } \Psi(\mathrm{d_{jm}}) = \mathrm{vm_{im}}, \quad \{\mathrm{vm_{i1}, vm_{i2}, \dots, vm_{im}}\} \subseteq \mathrm{VM}$$

the other datasets, $D - D_{fixed}$, are called flexible.

We consider all the workflow tasks are flexible and there are no fixed tasks because moving computation task to datasets is often cheaper than moving datasets to computation task nodes. To define a good measurement to compare separation between virtual machines, data interdependency within and between virtual machines are defined as follows:

Definition 2.6 (Within-VirtualMachine Data Interdependency $\mathbf{VMD_W}$).

$$VMD_W(\Psi) = \sum_{i=1}^{I} \sum_{\substack{\Psi(d_{j1}) = vm_i \\ \Psi(d_{j2}) = vm_i}} dp(d_{j1}, d_{j2}) \tag{4}$$

where $\mathrm{dp}(\mathrm{d_{j1}}, \mathrm{d_{j2}})$ is the data interdependency between task $\mathrm{d_{j1}}$and $\mathrm{d_{j2}}$, I is the maximum number of virtual machines in the Cloud.

Definition 2.7 (Between-VirtualMachine Data Interdependency $\boldsymbol{VMD_B}$).

$$VMD_B(\Psi) = \sum_{i_1 \neq i_2}^{\mathrm{I}} \sum_{\substack{\Psi(d_{j1}) = vm_{i_1} \\ \Psi(d_{j2}) = vm_{i_2}}} dp(d_{j1}, d_{j2}) \tag{5}$$

To achieve the data placement goal, BDAP uses heuristic information for its search direction of finding the best data placement scheme. Heuristic information should consider both within and between virtual machine interdependency. The heuristic is defined in BDAP as follows:

Definition 2.8 (Data Interdependency Greedy DG). The DG heuristic biases BDAP to select the data placement scheme with higher data interdependency. It is defined as:

$$DG(\Psi) = \frac{VMD_W(\Psi) + 1}{VMD_B(\Psi) + 1} \tag{6}$$

In this formula, the numerator measures Within-VirtualMachine Data Interdependency and the denominator measures the Between-VirtualMachine Data Interdependency. The bias 1 is set to avoid divided-by-zero in the case that the data interdependency between virtual machines get zero. A good data placement scheme has a higher DG. Therefore the output of BDAP is a data placement scheme with the highest DG.

In our system model we consider two types of system constraints in terms of data which are defined as follows:

Definition 2.9 (Data Placement Scheme Legality Constraints). Two types of illegal data placement schemes are considered in BDAP:

1) Virtual machine storage capacity constraint: The total amount of placed datasets into a virtual machine should be less than the available storage capacity of the virtual machine as it is impossible to fit all those datasets into the same virtual machine.

2) Non-replication constraint: Once a dataset is placed into a specific virtual machine, it is not allowed to place it into another virtual machine as data and task replication is not in the scope of this version of BDAP.

Definition 2.10 (Data Placement Solution). The data placement solution for scientific workflow, W, to execute in a Cloud computing environment, C, is to select a data placement scheme $\Psi \in \mathrm{P}$ to minimize the workflow communication cost (WCC) under the virtual machine storage capacity and non-replication constraints. In the next section, we explain our data placement strategy, BDAP, in detail.

III. PROPOSED DATA PLACEMENT ALGORITHM

The main steps of BDAP are depicted in Figure III. In the first phase, BDAP applies a metaheuristic optimization algorithm to place the original data. The main goal is to minimize workflow communication cost by minimizing the data movement between virtual machines in the Cloud within running a workflow. The main steps of this phase which applies in design-time are shown in the flowchart of Figure III. BDAP starts with calculating the data interdependency matrix. Then, it generates a set of legal data placement schemes randomly and calculates their heuristic values. In the following, for each data placement scheme, BDAP applies three main operators, Selection, Crossover, and Mutation sequentially to generate possibly better schemes with higher heuristic values. At the end of the algorithm, the best observed data placement scheme is recorded in Ψ_{best} and will be returned as the output of BDAP. Selection, crossover and mutation operators are defined as follows:

Definition 3.1 (Selection *SE*). Selection is the process of choosing two schemes for recombination and generation two new schemes. There are many methods to perform selection. We use the Roulette Wheel Selection techniques for BDAP. In this selection operator, the probability to choose a certain scheme is proportional to its heuristic value.

Definition 3.2 (Crossover *CO*). This operator combines two selected schemes to reproduce two new schemes. The idea is that the new generated schemes may be better and have higher heuristic value if they take the best characteristics from their parent schemes. For instance, suppose $\Psi_{\mathrm{l1}} < 1,2,1,2,3 >, \Psi_{\mathrm{l2}} < 2,2,1,3,1 >$ and the selected row number to crossover is 3 then $\Psi'_{\mathrm{l1}} < 1,2,1,3,1 > and\ \Psi'_{\mathrm{l2}} < 2,2,1,2,3 >$.

Definition 3.3 (Mutation *MU*). After crossover, BDAP applies mutation operator to an individual scheme to generate a new version of it such that a virtual machine position in the scheme have been randomly changed. Mutation prevents BDAP to be trapped in a local maximum heuristic value. For example, suppose $\Psi_{\mathrm{l}} < 1,2,1,2,3 >$ and the select row number is 4 and generated randomized number for position 4 is 3 then $\Psi'_{\mathrm{l}} < 1,2,1,3,3 >$.

For applying data placement strategy and analyzing the data interdependency, the whole workflow has to be designed. It means all tasks and datasets of the scientific workflow have to be specified. The BDAP algorithm is outlined in Algorithm I.

In the first step, BDAP generates *popsize* number of feasible and valid data placement schemes randomly with the locations for fixed-location datasets fixed. It also calculates the heuristic value of each individual scheme (lines1-5). The position numbers of the fixed-location datasets in the generated data placement scheme are fixed and will not change through the whole algorithm. In the next steps, BDAP applies three main operators to generate new schemes with a hopefully higher heuristic values until it reaches the max number of iterations.

```
Algorithm I. Big Data Placement (BDAP) – Original Datasets
Input:
    D: set of workflow datasets,
    DP: data interdependency matrix,
    popsize: size of population ,
    er: rate of elitism,
    cr: rate of crossover,
    mr: rate of mutation,
    num_iteration: number of iterations,

Output:
    The best data placement scheme, $\Psi_{best}$

1.  Begin
2.     for i = 1 to popsize do
3.        $\Psi$ ← Generate a legal data placement scheme randomly;
4.        $Pop \leftarrow < \Psi,\ DG(\Psi) >$;
5.     end for
6.     idx = 0;
7.     while ( idx ≤ num_iteration ) do
8.        ne = popsize × er; // number of elitism
9.        $Pop_E$ ← The best ne data placement schemes in Pop;
10.       nc = popsize * cr; // number of crossover
11.       for i =1 to nc do
12.           randomly select two data placement scheme $\Psi_A$ and $\Psi_B$ from Pop;
13.           generate $\Psi_C$ and $\Psi_D$ by one-point crossover for flexible
              datasets of $\Psi_A$ and $\Psi_B$;
14.           $Pop_C \leftarrow < \Psi_C, DG(\Psi_C) >$;
15.           $Pop_C \leftarrow < \Psi_D, DG(\Psi_D) >$;
16.       end for
17.       nm= popsize × mr;// number of mutation
18.       for i =1 to nm do
19.           select a data placement scheme $\Psi_j$ from $Pop_C$;
20.           $\Psi_j'$ ← mutate randomly a flexible virtual machine position
              number in $\Psi_j$ ;
21.           if $\Psi_j'$ is illegal
22.               update $\Psi_j'$ with a data placement scheme by repairing $\Psi_j'$;
23.           end if
24.           $\Psi_j \leftarrow \Psi_j'$;
25.       end for
26.       Pop ← $Pop_E$ and $Pop_C$;
27.       idx = idx +1;
28.    end while
29.    return the best data placement scheme $\Psi_{best}$;
30. end
```

First, it selects $ne = popsize \times elitism\_rate$ number of scheme with the highest heuristic value and saves them in the Pop_E (lines 9-10), these high-value schemes will transfer directly to the next generation of schemes to guarantee the convergence of BDAP. We apply the fitness proportionate selection, roulette wheel selection, for this step. The idea behind the roulette wheel selection technique is that each scheme is given a chance to select in proportion to its heuristic value. Then, it applies the crossover function and computes the heuristic value of the new generated schemes (lines 11-16). In the last step, BDAP applies the mutation operator for a randomly selected scheme along with computing its heuristic value (lines 17-25). In the crossover and mutation phases, BDAP does not change the number of virtual machine position for the fixed-location datasets and applied those function only on flexible datasets.

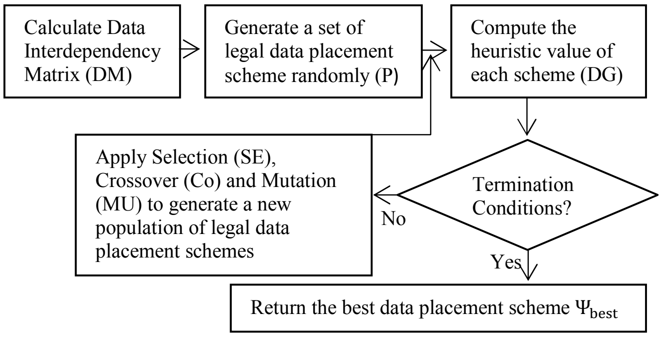
Calculate Data Interdependency Matrix (DM)
Generate a set of legal data placement scheme randomly (P)
Compute the heuristic value of each scheme (DG)
Apply Selection (SE), Crossover (Co) and Mutation (MU) to generate a new population of legal data placement schemes
No
Termination Conditions?
Yes
Return the best data placement scheme Ψ_{best}

Figure III. Flowchart of BDAP.

The idea behind the roulette wheel selection technique is that each scheme is given a chance to select in proportion to its heuristic value. Then, it applies the crossover function and computes the heuristic value of the new generated schemes (lines 11-16). In the last step, BDAP applies the mutation operator for a randomly selected scheme along with computing its heuristic value (lines 17-25). In the crossover and mutation phases, BDAP does not change the number of virtual machine position for the fixed-location datasets and applied those function only on flexible datasets. These three operators apply to the schemes till it reaches a certain number of iterations, a parameter defined by the user at the beginning of the algorithm. In the last step, the best data placement scheme Ψ_{best} is returned as the output of BDAP.

IV. EXPERIMENTS AND DISCUSSION

In this section, we present and discuss the simulation results and compares BDAP with the most competitive and Random approaches.

A. Simulation Setting

To evaluate performance of our proposed data placement approach, BDAP, we compare it with Yuan's work and random strategies. Yuan's work is the one of the most competitive algorithms in this field. It is a K-means based clustering algorithm which applies a heuristic binary clustering algorithm to precluster datasets into their appropriate virtual machines. Then, it greedily assigns the workflow tasks to each virtual machine such that it stores the most of its input dataset. Once an intermediate dataset is generated, it places it to the virtual machine that has the most interdependent datasets with the newly generated dataset.

We simulate a Cloud computing environment on the Wayne State University's high performance Grid Computing. We use eight grid computation nodes along with total storage capacity of 100 GB and compared the three algorithms by simulating a variety of real and synthetic workflows. We test BDAP using five synthetic workflow applications based on real scientific workflows [3]: Montage, CyberShake, Epigenomics, LIGO and SIPHT (Figure IV). These workflow applications are developed through the Pegasus workflow management system for different research domains like bioinformatics and astronomy. We select the large-size of each workflow with about 1000 number of tasks and assume each task can be executed on every virtual machines.

For our experiments, we run 100 times each of the selected workflows along with assigning five different numbers of datasets to their tasks randomly. The numbers of datasets are 5, 10, 25, 50 and 100, and dataset sizes are uniformly distributed in the range of [1TB, 100TB]. In addition, we consider five size numbers of virtual machines, 5, 10, 15, 20 and 25 in the range of [200TB 1PB] of storage capacity (as shown in Table II). Virtual machines storage capacities are selected in a uniformly distributed manner too. We demonstrate the performance of our proposed data placement algorithm and Yuan and Random approaches in terms of the average of the workflow communication cost defined in section 3. In our experiments, we assume that the data transmission rates among all virtual machines are fixed. Table III shows the value of parameters using in BDAP.

We do our experiments for two different scenarios, one scenario with considering 20% of fixed-location datasets and the other one without considering fixed-location datasets and consider the average of it.

B. Results

Figure V shows the Workflow Communication Cost (WCC), in terms of hour by varying the number of datasets and fixing the number of virtual machines. WCC is increased by increasing the number of datasets in all three strategies. However, it can be seen clearly that our strategy reduces WCC compared to the other strategies. This results in greater improvement margin with more number of datasets.

In the next step we calculate WCC by varying the numbers of virtual machines and fixing the number of datasets (Figure VI). Although WCC is increased by increasing the number of virtual machines, the increasing rate of our strategy is slower than the others. This results in greater improvement margin with more number of virtual machines.

We demonstrate performance of BDAP in terms of workflow communication cost by varying the number of datasets and virtual machines for five different types of workflows. We compare the BDAP strategy with Yuan as well as with random strategies. The result shows that BDAP manages to decrease effectively workflow communication cost more than Yuan and Random approaches.

V. Implementation

In our DATAVIEW system [13], we integrated the scientific workflow engine subsystem with FutureGrid academic research cloud provider to automatically provision virtual machines to execute scientific workflows in the Cloud. We implemented bash scripts to automatically provision VMs by first creating new VM images in the FutureGrid framework through configuring both hardware and software stack. Workflows execution is transparent to our data scientists. They can just create and run any arbitrary workflow and the system deploys a set of virtual machines, datasets and moves workflow tasks to the corresponding virtual machine.

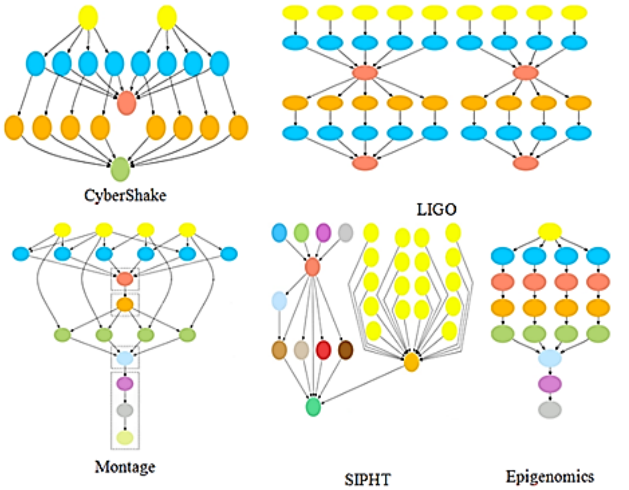

Figure IV: The structure of five realistic scientific workflows [3].

TABLE II. Description of dataset and virtual machine of the experiment.

Overall dataset and virtual machine	
# of datasets	[5,10,25,50,100]
Dataset size	1TB – 100TB
# of virtual machines	[5,10,15,20,25]
Virtual machines storage capacity	200TB – 1PB

TABLE III. Default setting for BDAP algorithm.

Overall dataset and virtual machine	
Population size	100
Initial population	Randomly generation
Maximum generation	100
Crossover probability	0.8-0.9
Mutation probability	0.3-0.5
Maximum iteration	1000

In design time, we created the sophisticated XML parser to parse the workflow specification, which is stored in the XML format. The XML parser extracted all workflow tasks, a set of input data products and a set of output datasets that will be generated at run time. The XML parser generated output (dpID, taskID) key/value pairs that contain mapping details to map datasets to corresponding workflow tasks. BDAP algorithm validate the (key, value) pairs to identify the optimal mapping of datasets and workflow tasks to the corresponding virtual machines.

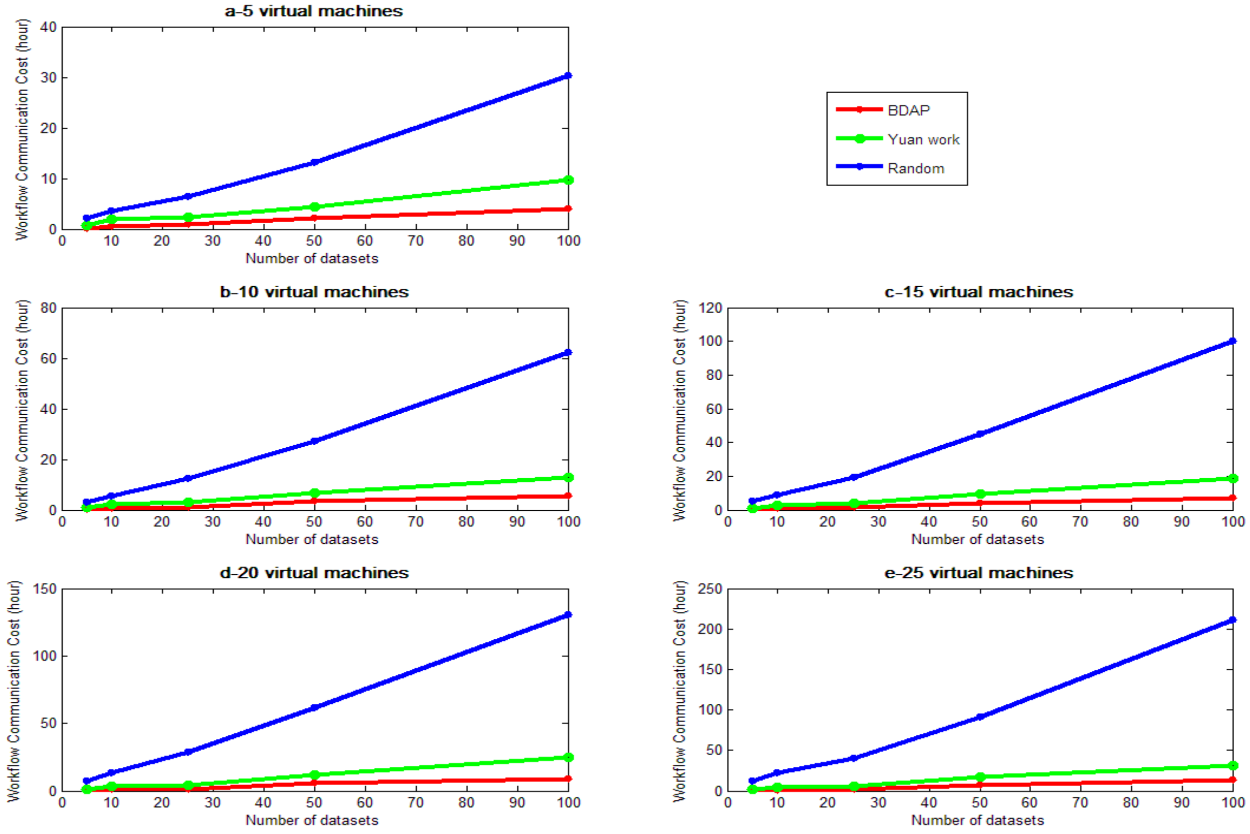

Figure V. Workflow Communication Cost by varying the number of datasets.

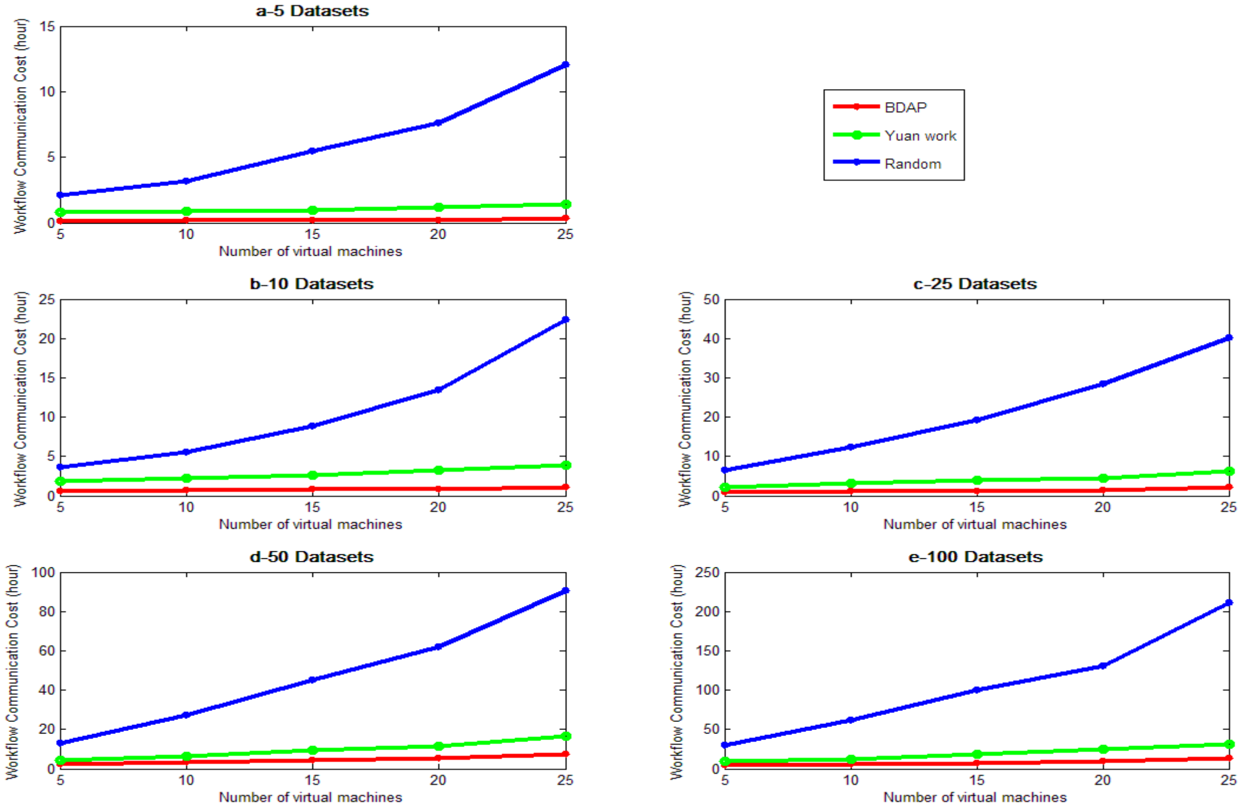

Figure VI. Workflow Communication Cost by varying the number of virtual machines.

In running time, DATAVIEW provisioned a set of virtual machines in FutureGrid and deployed datasets to the corresponding virtual machines based on the output of BDAP. In our DATAVIEW system, we used files as a dataset type and used SCP command to move actual files from our DATAVIEW system to the provisioned virtual machines. In the next step, we assigned all the workflow tasks to the provisioned virtual machines. After assigning workflow tasks and datasets, the workflow was executed and intermediate datasets were moved to the corresponding virtual machines. Data flow between each workflow task was implemented by the SCP command. The final dataset was moved from its virtual machine to the DATAVIEW system and the results were published to the user. In this way, all low-level details were hidden from the data scientists and only the intermediate and final data products generated by the workflow were visible to data scientists. Table IV shows some of the result of applying BDAP for the execution of workflow in Example I.

TABLE IV. SOME RESULT OF BDAP RUNNING.

The best data placement scheme in the	
First population <d#, vm#>	**Last (10^{th}) population <d#, vm#>**
$<d_1,vm_1><d_2,vm_2><d_3,vm_3><d_4,vm_3><d_5,vm_1><d'_1,vm_2><d'_2,vm_1><d'_3,vm_1><d'_4,vm_3>$ DG = 0.1671 and WCC = 0.0097 hr	$<d_1,vm_3><d_2,vm_1><d_3,vm_3><d_4,vm_1><d_5,vm_2><d'_1,vm_1><d'_2,vm_2><d'_3,vm_1> <d'_4,vm_2>$ DG = 3.4032 and WCC = 0.0041 hr
$<d_1,vm_2><d_2,vm_1><d_3,vm_3><d_4,vm_1><d_5,vm_3><d'_1,vm_3><d'_2,vm_1><d'_3,vm_2> <d'_4,vm_3>$ DG = 0.2513 and WCC = 0.0083 hr	$<d_1,vm_1><d_2,vm_2><d_3,vm_1><d_4,vm_2><d_5,vm_3><d'_1,vm_2><d'_2,vm_1><d'_3,vm_3> <d'_4,vm_2>$ DG = 3.4678 and WCC = 0.0033 hr
$<d_1,vm_2><d_2,vm_2><d_3,vm_1><d_4,vm_3><d_5,vm_3><d'_1,vm_3><d'_2,vm_3><d'3,vm_1> <d'_4,vm_3>$ DG = 0.3165 and WCC = 0.0081 hr	$<d_1,vm_2><d_2,vm_2><d_3,vm_1><d_4,vm_2><d_5,vm_3><d'_1,vm_2><d'_2,vm_3><d'_3,vm_3> <d'_4,vm_1>$ DG = 3.3692 and WCC = 0.0042 hr

VI. RELATED WORK

Previous research studies for distributed computing environment have been mainly focused on the performance modeling and optimization of job scheduling and task allocation. But due to the rapid increase in the size of available data over the internet and the emerging field of Big Data, data placement becomes a fundamental spot in the Cloud recently. Kosar et al., [12] proposed a framework for distributed computing systems, which considered the data placement subsystem as an independent module along with computation subsystems. In their proposed model, data placement jobs can be queued, scheduled, monitored, managed and even checkpointed. Kayyoor et al., [11] considered the data placement and replication problems together for the distributed environments. They claimed minimizing of query latencies is not a critical issue in many scenarios of analytical workloads, so they tried to minimize the average number of using computation nodes by grouping the most interdependent data together based on their occurrences of the common query accesses. Chervenak et al., [5] explored the advantages of separation of data placement as a service from workflow management systems. By applying an autonomous data placement service along with data replication service, they evaluate and display the benefits of pre-staging data compare to the data stage in and out strategies of the Pegasus workflow management systems. However, none of the above studies decrease the data movement among virtual machine in the Cloud.

Some of the workflow management systems have been extended to execute scientific workflow in Clouds. Pegasus is designed to execute scientific workflows on a number of distributed resources such as local machines, clusters or Cloud. Nimbus is an integrated set of tools which allows scientific users to deploy a cluster into infrastructure Clouds to execute their scientific workflows. Eucalyptus is an open source Cloud management software to create on-demand, self-service private Cloud resources.

In Catalyurek et al., [4] workflows were modeled by the hypergraph concept and a hypergraph portioning technique, k-way partitioning, is applied to minimize the cutsize. In that way, they cluster the workflow tasks as well as their required data in the same execution site. One of the closest works to our strategy is Yuan et al., [14], in which they applied a greedy binary clustering to precluster datasets; then they greedily assigned the workflow tasks to an execution site that contains the most of the input datasets. At the end, once an intermediate dataset was generated, they placed it to the execution site which has the most interdependent dataset. Although their approach placed the most interdependent dataset together and can reduce data movement, the algorithm is greedy and it clustered the data dependency matrix into two parts in each iteration and so their clustering technique was sensitive to the selection point in any iteration. The other close work to our study is Er-Dun et al., [8] in which they applied a genetic algorithm to find their data placement solution along with load balancing factor. Their approach reduced data movement, however, they did not consider data interdependency between virtual machines and also they did not consider task assignment.

VII. CONCLUSIONS AND FUTURE WORK

We proposed BDAP, a data placement strategy for Cloud-based scientific workflows. BDAP minimized the total amount of data movement between virtual machines during the execution of the workflows. Our extensive experiments and comparisons indicated that BDAP outperformed other proposed algorithms on minimizing data movement. Scientific workflows consume and produce huge datasets. Applying data replication can reduce data movement as well. So in future work, we plan to improve BDAP by applying data replication techniques. In addition, we considered data placement for executing of a single workflow. However, in real world, multiple workflows can be executed concurrently. Therefore, we plan to extend BDAP strategy to achieve data placement for the execution of multiple workflows simultaneously.

ACKNOWLEDGMENT

This work is supported by National Science Foundation, under grant NSF ACI-1443069. In addition, this material is based upon work supported in part by the National Science Foundation under Grant No. 0910812.

REFERENCES

[1] D. Agrawal, A. E. Abbadi, S. Antony, and S. Das, "Data management challenges in cloud computing infrastructures." In Databases in Networked Information Systems, pp. 1-10, 2010.

[2] J. Wang, D. Crawl, I. Altintas, W. Li, "Big Data Applications Using Workflows for Data Parallel Computing," Journal of Computing in Science and Engineering, vol. 16, no. 4, pp. 11-21, 2014.

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

[4] U. V. Catalyurek, K. Kaya and B. Ucar, "Integrated Data Placement and Task Assignment for Scientific Workflows in clouds," In Proceedings of the fourth international workshop on Data-intensive distributed computing, pp. 45-54, 2011.

[5] A. Chervenak, E. Deelman, M. Livny, M. Su, R. Schuler, S. Bharathi, G. Mehta, and K. Vahi, "Data Placement for Scientific Applications in Distributed Environments," In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, pp. 267-274, 2007.

[6] E. Deelman and A. Chervenak, "Data Management Challenges of Data-Intensive Scientific Workflows," In Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on, pp. 687-692, 2008.

[7] Y. Zhao, I. Raicu, X. Fei and S. Lu, "Opportunities and Challenges in Running Scientific Workflows on the cloud," In Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on, pp. 455-462, 2011.

[8] Z. Er-Dun, Q. Yong-Qiang, X. Xing-Xing and C. Yi, "A Data Placement Strategy Based on Genetic Algorithm for Scientific Workflows," In Computational Intelligence and Security (CIS), 2012 Eighth International Conference on, pp. 146-149, 2012.

[9] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," In Grid Computing Environments Workshop, 2008. GCE'08, pp. 1-10, 2008.

[10] G. Juve and E. Deelman, "Scientific workflows and clouds," Journal of Crossroads, vol. 16, no. 3, pp. 14-18, 2010.

[11] A. K. Kayyoor, A. Deshpande and S. Khuller, "Data Placement and Replica Selection for Improving Co-location in Distributed Environments," arXiv preprint arXiv:1302.4168, 2013.

[12] T. Kosar and M. Livny, "A Framework for Reliable and Efficient Data Placement in Distributed Computing Systems," Journal of Parallel and Distributed Computing (JPDC), Vol.65, no. 10, pp. 1146-1157, 2005.

[13] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua and F. Fotouhi, "Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System," Proceedings of the IEEE International Conference on Services Computing (SCC), pp. 335-342, 2008.

[14] D. Yuan, Y. Yang, X. Liu and J. Chen "A data placement strategy in scientific cloud workflows," Future Generation Computing Systems 26, no. 8, pp. 1200-1214, 2010.

[15] D. Yuri, P. Membrey, P. Grosso and C. Laat "Addressing Big Data Issues in Scientific Data Infrastructure," In Collaboration Technologies and Systems (CTS), 2013 International Conference on, pp. 48-55, 2013.