A Big Data Placement Strategy in Geographically Distributed Datacenters

Laila Bouhouch
National School of Computer science and Systems Analysis, Mohamed V University
Rabat, Morocco
lailabouhouch94@gmail.com
laila.bouhouch@mines-paristech.fr

Mostapha Zbakh
National School of Computer science and Systems Analysis, Mohamed V University
Rabat, Morocco
m.zbakh@um5s.net.ma

Claude Tadonki
Mines ParisTech-PSL Centre de Recherche en Informatique (CRI)
Paris, France
claude.tadonki@mines-paristech.fr

***Abstract*—With the pervasivness of the "Big Data" characteristic together with the expansion of geographically distributed datacenters in the Cloud computing context, processing large-scale data applications has become a crucial issue. Indeed, the task of finding the most efficient way of storing massive data across distributed locations is increasingly complex. Furthermore, the execution time of a given task that requires several datasets might be dominated by the cost of data migrations/exchanges, which depends on the initial placement of the input datasets over the set of datacenters in the Cloud and also on the dynamic data management strategy. In this paper, we propose a data placement strategy to improve the workflow execution time through the reduction of the cost associated to data movements between geographically distributed datacenters, considering their characteristics such as *storage capacity* and *read/write speeds*. We formalize the overall problem and then propose a data placement algorithm structured into two phases. First, we compute the estimated transfer time to move all involved datasets from their respective locations to the one where the corresponding tasks are executed. Second, we apply a greedy algorithm in order to assign each dataset to the optimal datacenter w.r.t the overall cost of data migrations. The heterogeneity of the datacenters together with their caracteristics (storage and bandwith) are both taken into account. Our experiments are conducted using Cloudsim simulator. The obtained results show that our proposed strategy produces an efficient placement and actually reduces the overheads of the data movement compared to both a random assignment and a selected placement algorithm from the litterature.**

***Index Terms*—Big Data, Cloud Computing, Data Placement, Greedy algorithm, Cloudsim.**

I. Introduction

Nowadays, various scientific fields like physics [2], astronomy [3], bio-informatics [4], earthquake science [5] and social media [6], produce huge amounts of data every day. Data are generated either from physical devices such as those of the IoT or they are produced as the output of specific applications. This data explosion is generally known through the generic word " Big Data". Hence, (data) scientists need to develop new methodologies for the analyzis, storage and process of a huge amount of data in the most efficient way. Typically, Cloud computing appears as the best infrastructure to run this type of applications and also to improve their performance [7].

Since their emergence around 2007 [8], cloud-based technologies since have been used successfully in many areas [10; 11; 12; 13] by offering diverse services such as scalable infrastructures, elastic provisioning, on-demand resources and distributed computing [9]. The benefits of Cloud computing can be summarized in three main points: 1) provide desired massive storages and high-performance computing resources; 2) allocate on-demand resources easily; 3) construct an infrastructure at a lower cost made possible by the availability of several datacenters.

Based on the Internet, cloud paradigm continues to grow in such a way that some cloud providers use tens of geographically distributed datacenters around the globe [14]. In addition, this growth offers more functionalities for scientists to easily configure and collaborate together. Further, it provides many appropriate frameworks for processing "Big Data" applications as well as efficiently distribute and cluster tasks and data over distant heterogeneous sites. This heterogeneity is one of the characteristics of each datacenter and refers to storage capacity and processing speed [15]. However, on such type of platforms, the task execution time and the overhead of data movements both depend on the target machine.

For a large-scale and distributed applications executed across different datacenters, a single task might require several pieces of data (dataset) spread over the cloud system [16]. When running such applications, since some dataset won't be available on the same node where a consumer task is assigned, data movements will occur and the associated time overhead might be significant, especially in a "Big Data" context. Thus, data placement in a cloud system with several distant datacenters is a crucial aspect related to the global time/energy performances [1].

To deal with the previously mentioned issue, we propose an efficient data placement strategy for data-intensive applications deployed in Cloud environment. The main goal is placing the datasets carefully in order to reduce the overall cost of the data transfers, not only considering datacenters heterogeneity regarding their characteristics like the read/write speeds, but also the limited bandwidth of the links among the datacenters. Technically, to reduce data movements among datacenters, we first compute a cost matrix which gives the total transfer time of each dataset considering each source location and all target locations (those of the tasks that use the dataset). Afterwards, using that matrix, we apply the greedy algorithm to optimally choose the assignment of the datasets to the datacenters. More precisely, for a given task, all its required datasets will be moved in an efficient time from their initial location to where the execution of this consumer task is assigned to.

The rest of the paper is organized as follows: Section 2 discusses some related works. In section 3, we formalize our data movement model. Section 4 is devoted to explain in details the proposed data placement strategy. Section 5 depicts and discusses the experimental results using the Cloudsim tool. Finally, we draw a conclusion and indicate some future works in Section 6.

II. Related Work

Optimal datasets placement is one of the issues of the management of geographically distributed datacenter systems. A reasonable data placement allows to effectively reduce the time/energy overhead of data movements. There are many disseminated contributions proposing different algorithms and techniques related to that problem. We comment some of them in this section.

To solve the data placement problem, authors of [17] consider a K-means algorithm. They compute a dependency matrix, then use it to investigate how to distribute the data over the datacenters. The datasets that are strongly interdependent are placed within the same datacenter. In [18], the authors establish a data placement strategy based on a genetic algorithm and compare it to other algorithms such as a Monte Carlo algorithm and an exhaustive search algorithm. The results of the proposed approach gives better nearly-optimal solutions for the data placement and data scheduling among datacenters. Another study is described in [19], which proposes two contributions: (1) a heuristic approach following the genetic algorithm paradigm (2) a load balancing among the datacenters considering their capacity constraints and workloads. This combination gives better schedulings and extends the search space. Authors of [20] propose a new way to cluster the datasets based on the correlation between pairs of datasets following the question of how much the amount of data transfers would be if these two datasets were stored within two distinct datacenters. While calculating this correlation, they incorporate the factor orelated to the size of the dataset that will yield the lowest impact. In [21], the authors suggest a data placement strategy to minimize the total amount of data transfers between virtual machines. They compute the interdepencies between pairs of datasets and the most related are stored on the same virtual machine following a genetic algorithm. They also consider the data transfer rate between virtual machines to decrease the data movements.

However, to our knowledge, most of the previously mentioned works try to reduce the data movements mainly, rather focusing on the time consumption of the data transfers, which is the genuine concern.

In fact, in a heterogeneous Cloud Computing environment, every datacenter has its own capacity and speeds (processing and read/write). Furthermore, while executing a data-intensive application among geographically distributed datacenters, several datasets are migrated from one datacenter to an another distant one. Hence, the overall cost of the data transfers rates will highly affect the application performance, and this strongly depends on the interconnection bandwidth. So, only decreasing number of data movements over datacenters is not equivalent to reducing the data transfers time cost. This is the reason why in our work we propose a data placement strategy in order to decrease time to transfer the datasets when running "Big Data" applications in a Cloud environment and henceforth improve the overall performance.

III. Modeling of data placement system

Aiming at reducing the cost of the data transfers among datacenters in a Cloud environment, we look for a distribution of the datasets among datacenters such that overall data transfer time is the optimal. To address this data placement problem, we are first going to describe the elements of our model. Table I provides all necessary symbols and notations.

A. Datecenters graph

To model our problem, we consider a system of geographically distributed but interconnected datacenters. The problem is modeled by using a directed Acyclic graph $G = (M, w, r, B)$. M is the set of nodes (datacenters) expressed by $\{m_i, 1 \leq i \leq p\}$, where m_i is a single datacenter. Each datacenter m_i is characterized by a read speed r_i and a write speed w_i. Finally, B is the set of edges connecting the datacenters where each edge $\{b_{ij}, 1 \leq i, j \leq p\}$ is the transfer speed between m_i and m_j (∞ if there is no link). We logically set $b_{ii} = 0$. The matrix $B = (b_{ij})$ is the interconnection matrix between the datacenters.

B. Initiate tasks

In our proposed model, we assume that the Cloud system receives a fixed number of tasks. Tasks have different sizes/lengths and may require several datasets for their execution. As indicated in Table I, we denote the set of task as $T = \{t_1, t_2, \ldots, t_n\}$, where n is the number of tasks to execute. We consider that each task is assigned to a specific datacenter for execution. For that, we define a vector $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, so that t_i is executed in m_{α_i}. Several tasks can be executed at the same time, but each one is assigned to only one datacenter.

C. Assign data to tasks

We define D as a set of all datasets, where each dataset d_i, $1 \leq i \leq m$, has volume v_i.

A task may require one or multiple datasets and a single dataset might be consumed by many tasks. This lead us to create the $m \times n$ matrix F describing the mapping between tasks and data, in other words :

$$f_{ij} = \begin{cases} 1 & \textit{if } d_i \in D \textit{ is required by } t_j \in T \\ 0 & \text{otherwise} \end{cases}$$

TABLE I
SYMBOLS AND NOTATIONS

Denotations	
M	Set of the datacenters
m_i	Designation of the i^{th} datacenter in M
c_i	Storage capacity of datacenter m_i
r_i	Read speed of datacenter m_i
w_i	Write speed of datacenter m_i
b_{ij}	Bandwidth between two datacenters, m_i and m_j
T	Set of the tasks
t_i	Designation of the i^{th} task in T
α	Array of tasks assignement (i.e task t_i is assigned to m_{α_i})
α_i	Task t_i is executed in datacenter m_{α_i}
D	Set of datasets
d_i	Designation of i^{th} dataset in D
v_i	Volume of dataset d_i
F	Matrix to map datasets and tasks
f_{ij}	$f_{ij} = 1$ means d_i is required by t_j, 0 otherwise
τ	Transfer time matrix
τ_{ik}	Total time needed to transfer d_i from datacenter m_k to every datacenter where there is task that uses d_i as input
ϕ	Array of the final datasets assignments
ϕ_i	Dataset d_i is assigned to datacenter m_{ϕ_i} (output of the placement algorithm)

D. Constraints

Two types of data placement constraints are considered:

1) Datacenter storage capacity constraint: The total size of stored data should not exceed the available storage capacity of the datacenter.

2) Non-replication constraint: A datatset is only placed into one specific datacenter and no data replication is considered (this aspect is a potential extension of this work).

E. Data placement solution

Let us consider a vector $\Phi \in N^{|D|}$ where $\phi_i, i \in 1, \dots, m$ is the index of datacenter housing d_i, thus:

$$d_i \text{ is placed in datacenter } m_{\phi_i} \tag{1}$$

The data placement solution aims to minimize the data transfers cost when moving datasets from one location to other, considering the mentionned constraints above. In the next section, we describe more clearly our propsed data placement strategy.

IV. Process of the proposed data placement algorithm

In this section, we explained the main steps of our data placement algorithm in order to minimize communication cost while moving datasets among datacenters. The proposed algorithm resolves the aforementioned problems and is composed on two main steps. As a first step, we compute the total transfer time of the datasets between the datacenters based on their volume and read/write bandwidths. In the second step, we apply a greedy algorithm to assign the datasets to the datacenters based on the computed potential transfer time and the storage capacities of the nodes.

As we know, a given dataset might be stored in the same datacenter where (one of) its consumer task is executed or in a another (distant) one. No transferring is needed in the former, while the latter implies a data migration.

Let assume that a given dataset d_i is stored in datacenter m_j and is used by a task executed on datacenter m_k $(j \neq k)$. The time needed to transfer d_i from its source m_j to the destination m_k, $1 \leq j, k \leq p$ and $1 \leq i \leq m$, is expressed as follows :

$$\tau_{jk}^{(i)} = v_i \times (\frac{1}{r_j} + \frac{1}{w_k} + \frac{1}{b_{jk}}) \tag{2}$$

Since a dataset d_i might be required by many tasks (potentially all of them), its total transfer time from m_j considering all requests during the whole execution can be written as follow :

$$\begin{aligned}\tau_{ij} &= \sum_{k=1}^{|T|} f_{ik} \times \tau_{j\alpha_k}^{(i)} \\ &= \sum_{k=1}^{|T|} f_{ik} (\frac{1}{r_j} + \frac{1}{w_{\alpha_k}} + \frac{1}{b_{j\alpha_k}}) \times v_i \end{aligned} \tag{3}$$

Following equation (3), we can compute the aggregated migration time matrix $\tau = (\tau_{ij})$, where $1 \leq i \leq m$ and $1 \leq j \leq p$.

A. Compute τ matrix

Algorithm 1 depicts the pseudocode for computing the data transfer time matrix as explained in the precedent section. To apply our data placement strategy, all of the datacenters, tasks and datasets are the inputs.

For each data d_i, we go through each datacenter m_j in order to estimate the necessary time τ_{ij} that represents the overall time of moving the dataset d_i from the datacenter m_j to every location α_k where there is one of its consumer task t_k (line 9 to 19).

τ_{ij} in line 15 is calculated using the data size v_i, the read speed r_j of the datacenter m_j that we initially consider as the potential location of dataset d_i, the write speed w_{α_k} of the datacenter m_{α_k} where d_i could be moved for the execution of task t_k, and finally the bandwidth between the two datacenters represented by $b_{j\alpha_k}$.

Once matrix τ is created, we sort each of its rows (line 20) in ascending order and store the corresponding permutation into an array σ_i. In other words, $\sigma_i(j)$ is the index of the datacenter that yields the j^{th} transfer time cost following an ascending order. The goal is to get the best possible location (datacenter), starting with the one that gives the smallest transfer time.

As an example, we consider 4 datasets (d_1, d_2, d_3, d_4), 3 datacenters (m_1, m_2, m_3) and 3 task (t_1, t_2, t_3), with $\alpha_1 = 1$, $\alpha_2 = 3$ and $\alpha_3 = 2$, which means that the tasks are executed in the datacenters as indicated by the sequence (m_1, m_3, m_2). The matrix F is given by:

$$F = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The first line denotes that d_1 is required by t_2 and t_3, the second one means that d_2 is required by t_1 and t_3 and so on.

Based on line 15 in Algorithm 1, we get the following τ matrix:

$$\tau = \begin{pmatrix} 5 & 3 & 4 \\ 4 & 7 & 2 \\ 3 & 5 & 4 \\ 6 & 1 & 2 \end{pmatrix}$$

$\tau(1,1)$ indicates total expected time to move d_1 from m_1 to α_2 and α_3 (resp. m_3, m_1), where its consumer tasks t_2 and t_3 are scheduled for their execution.

Afterwards, we sort τ matrix row by row as indicated in line 20 of Algorithm 1 and obtain new matrix σ provided below:

$$\sigma = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \\ 1 & 3 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

The first position in the first line, $\sigma(1,1) = 2$, implies that the least transfer time of d_1 compared to other datacenters in the system is obtained with m_2 and equals $\tau(1,2) = 3$ units of time. In contrast, the last position for the same data yields a transfer time of 5 ($\tau(1,5)$) from m_1, which is the highest cost.

B. Algorithm for an efficient placement

Algorithm 2 implements the proposed strategy to get an efficient placement of the datasets onto the datacenters so as to reduce the total transfer time. For this purpose, we apply a greedy algorithm which iteratively assigns each dataset to the most appropriate datacenter. Considering a variable c as an index representing the current column in the transfer matrix τ, a variable s to depict the number of selected datasets and a binary variable *placed* to indicate if a dataset is already assigned to a location or not.

Algorithm 1 Calculate data transfer time matrix

```
Input:
 1: $M = (m_1, m_2, \dots, m_p)$ : Set of datacenters
 2: $T = (t_1, t_2, \dots, t_n)$ : Set of tasks
 3: $D = (d_1, d_2, \dots, d_m)$ : Set of datasets
 4: $C = (c_1, c_2, \dots, c_p)$ : Capacities of the datacenters
 5: $V = (v_1, v_2, \dots, v_m)$ : Volumes of the datasets
 6: $F$ : Matrix of relationship between datasets and tasks
 7: $\tau$ : Matrix of datasets transfer times over the datacenters
Output:
 8: $\phi = (\phi_1, \phi_2, \dots, \phi_m)$ : Array of final data placement
    // Computation of $\tau$ matrix
 9: for $i \in D$ do
10:     for $j \in M$ do
11:         $\tau(i,j) \leftarrow 0$
12:         for $k \in T$ do
13:             if $f(i,k) \neq 0$ then
14:                 if $j \neq \alpha_k$ then
15:                     $\tau(i,j) \leftarrow \tau(i,j) + (\frac{1}{r_j} + \frac{1}{w_{\alpha_k}} + \frac{1}{b_{j\alpha_k}}) \times v_i$
16:                 end if
17:             end if
18:         end for
19:     end for
    /* sort elements of each row $\tau(i,:)$ in ascending order */
20:     $[\sigma(i,:)] \leftarrow sort(\tau(i,:))$    /* $\sigma_i(j) = k$, the $k^{th}$ element of $\tau(i,:)$ */
21: end for
```

As long as *s* is lower than the total number of datasets on the system and *c* does not exceed the total number of datacenters. (line 6)

- We firstly initialize both *max* and *k* to -1; (line 7-8)
- We examine the datasets individually and for each one:
  1) We verify if the dataset is already placed (line 10), if so we check for the next one. Otherwise, we choose the first column of $\sigma(i)$ returning the index ℓ, which is supposed to be the current best placement. Then, we check (line 12) if the transfer time is higher than the current maximum (max) and if the remaining capacity (c_ℓ) allows to consider the corresponding volume v_i.
  If both conditions are true, then we update max value with the current transfer time and k with the index of the current dataset.
  Otherwise max and k keep their previous values.
  2) The main goal of the code bloc from line 9 to line 17 is to go through all unplaced datasets and pick the one with the highest transfer time that fit in the chosen datacenter. Thereby, we avoid transition to the next position $\sigma(i+1)$, which will certainly cost more since we have considered an ascending order.
  The idea here is that, for a given datacenter (column id c), we look for the eligible dataset (that fullfil the capacity constraint) with the highest transfer time if stored onto datacenter number c. For that datacenter,

this is the lowest transfer time since the possibilities (row of the matrix τ) are sorted in an ascending order. This is a max/min procedure, which aim at placing the dataset with the higher potential transfer cost at the earliest.

- At the end of the iteration on the current column (for the corresponding datacenter), either we were able to select on dataset (i.e. $k \neq 1$) or we couldn't (mainly because the capacity constraint was always blocking).
  1) In the first case (line 18 to 23), the dataset number k will be placed on the datacenter number c (considering the sorting, thus $\sigma[k, c]$) instead of c). Thus we set $\phi_k = \ell$ (line 21), where $\ell = \sigma[k, c]$, indicates that the dataset k is already placed (line 20), and update the capacity of the selected datacenter (line 22). Finally, we increment the variable s to move to the next data.
  2) In the second case, where no compatible dataset was found, we just increase c to pass to the next column.
- After iterating over all the datasets and over all the datacenters, We check if ($s = m$), which means that the problem of data placement is successfully resolved and every dataset is assigned to a datacenter. Otherwise, a problem has been encountered (because of the capacity constraints) and we couldn't place some datasets.

Applying this steps to our previous illustrative example, we get the final data placement solution depicts in Figure 1.

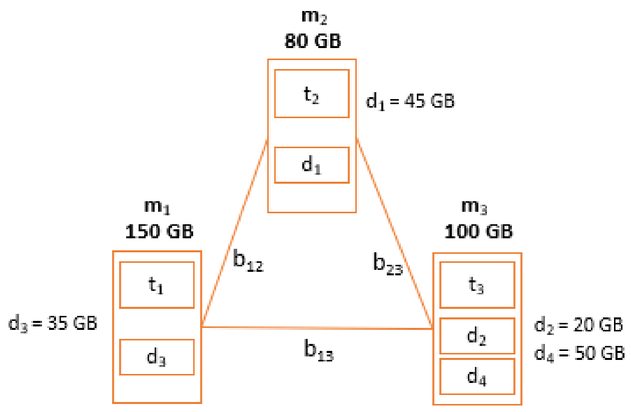

Fig. 1. A sample data placement solution

C. *Workflow communication cost*

Considering a worfklow W and a cloud system C, the workflow communication cost equals the total data transfer cost for executing the whole tasks of the workflow W in the Cloud C and can be estimated using formula 4.

$$WCC(W,C) = \sum_{j=1}^{|T|}\sum_{i=1}^{|D|} f_{ij} \times \tau^{(i)}(\phi_i, \alpha_j) = \sum_{j=1}^{|T|}\sum_{i=1}^{|D|} f_{ij}\left(\frac{1}{r_{\phi_i}} + \frac{1}{w_{\alpha_j}} + \frac{1}{b_{\phi_i \alpha_j}}\right) \times v_i \quad (4)$$

Algorithm 2 Algorithm for an efficient datasets placement

```
Input: /* Datasets placement following a greedy approach*/
 1: $c$: index of the current column in the transfer cost matrix
 2: $s$: number of already placed datasets
 3: $placed$: binary array indicating the selection status

 4: $c \leftarrow 1$
 5: $s \leftarrow 0$
 6: while ( $s < m$ & $c \leq p$ ) do
 7:     $max \leftarrow -1$
 8:     $k \leftarrow -1$
 9:     for $i \leftarrow 1$ to $m$ do
10:         if ($placed[i] \neq 1$) then
11:             $\ell \leftarrow \sigma[i,c]$ {id of our $c^{th}$ best datacenter choice for dataset $d_i$}
12:             if (($\tau(i,\ell) > max$) & ($c_\ell - v_i > 0$)) then
13:                 $max \leftarrow \tau(i,\ell)$
14:                 $k \leftarrow i$
15:             end if
16:         end if
17:     end for
18:     if ($k \neq -1$) then
19:         $\ell \leftarrow \sigma[k,c]$ {id of the best datacenter for dataset $d_k$}
20:         $placed[k] \leftarrow 1$
21:         $phi[k] \leftarrow \ell$ {$d_k$ will be stored in datacenter $m_\ell$}
22:         $c_\ell \leftarrow c_\ell - v_k$ {update of the capacity of datacenter $m_\ell$ as it receives dataset $k$}
23:         $s$++
24:     else
25:         $c$++
26:     end if
27: end while
28: if ($s = m$) then
29:     Data placed successfully !
30: else
31:     Problem with data placement !
32: end if
```

V. Experiments and discussion

A. *Simulation set-up*

We compare our proposed strategy for data placement with a random strategy and Yuan's work [17], which is one of the most known algorithms in this field. Yuan's solution is based on a K-means algorithm to create a data dependency matrix. Afterwards, the datasets are clustered based on the dependencies using a recursive binary partitionning so that datasets strongly interdependent are assigned to the same datacenter. The random strategy is based on a random assignment of the datasets to the datacenters.

To carry out our experiments, we simulate a Cloud computing environment using Cloudsim tool [22]. We create a workflow with about 1000 tasks with different workloads and assume that each task is scheduled in one datacenter. We limit the number of datacenters to 5, 10, 15 and 25. The storage capacities of datacenters are uniformly distributed in the range [1PB, 25PB]. Beside that, we randomly assign five different datasets to each task. The number of datasets are 5, 10, 25, 50, 75 and 100, and their sizes also uniformly distributed within the range [1TB, 100TB]. Overall description is shown in Table II. We run each scenario 100 times and take the mean value as the final result.

TABLE II
DEFAULT SETTINGS FOR OUR EXPERIMENT

Components & Values	
# of datasets	[5,10,25,50,75,100]
Dataset size	[1TB - 100TB]
# of datacenters	[5,10,15,20,25]
Datacenter capacity	[1PB - 25PB]

B. Simulation results

In our experiments, we evaluate our proposed data placement algorithm and compare the performances with those of Yuan strategy and a random assignment, using the average of the workflow communication cost (WCC) as defined in section IV-C as measurement.

Based on our previous work [23], we performed the workflow test 100 different times using the same parameters and then compute the average together with the mean value of the workflow communication cost.

In the first experiment, as shown in Figure 2, we set the number of geographically distributed datacenters to 5, 10, 15, 20 and 25 for the workflow execution and vary for each case the number of datasets from 5 to 100. We can observe that by increasing the number of dataset, WCC always increases in all cases. However, we can clearly see the obvious advantage of our greedy algorithm compared to k-means clustering and the random approach, we are always better regarding reduction of communication cost.

In the case of executing workflow tasks over 5 datacenters, our approach gives 2.42 hours on average, next to 4.42 and 7.14 hours on average for the k-means algorithm and random approach respectively. Thus, our proposed approach reduces effectively the cost of data movement by 45.12% and 66.10%, respectively. The results show an increasing progress with more and more datasets.

In the second experiment, we set the number of datasets to 5, 10, 25, 50, 75 and 100, change the number of datacenters in the range [5,25] and calculate WCC as shown in Figure 3. From that figure, we notice that WCC coutinuously increases with the number of datacenter in all data placement strategies. Despite this, the results show that our strategy is better than k-means and random strategies in reducing the total transfer time. For example, in the case 50 datasets and only varying the number of datacenters, the greedy based strategy reduces the cost by a rate of 66.52% and 91.60%, compared respectively to k-means and random approaches. Moreover, for the same case, our strategy gives 1.41 hour as a difference between the result of running the workflow on 5 datacenters and 25 datacenters. In the other hand, k-means and random give 12.67 and 63.95 respectively. Thereby, we can say that our proposed algorithm has a slower increasing rate than the others.

VI. Conclusion and Future work

This paper proposed a data placement strategy to address the problem of Big Data movements among geographically distributed datacenters in the Cloud environment. Our aim is to reduce the total cost of migrating datasets between datacenters during the execution of the tasks. Thus, we proposed an algorithm where we first create a matrix of data transfers costs then applied a greedy procedure to get the final assignment of the datasets to the datacenters. Compared with other data placement approaches, our experimental results show that our approach overcome the other algorithms in minimizing time consumption for moving the datasets. As a future work, we intend to consider load balancing to avoid overloaded datacenters, as well as data replication wich may increase the availability of datasets in Cloud environment.

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] Barnes, T. J., Wilson, M. W. (2014). “Big Data, social physics, and spatial analysis: The early years.” Big Data & Society. https://doi.org/10.1177/2053951714535365.

[3] Zhang, Zhao et al. “Scientific computing meets big data technology: An astronomy use case.” 2015 IEEE International Conference on Big Data (Big Data) (2015): 918–927.

[4] L. Wang, Y. Wang, Q. Chang, “Feature selection methods for big data bioinformatics: A survey from the search perspective”, Methods, Volume 111, 2016, Pages 21–31, ISSN 1046-2023, https://doi.org/10.1016/j.ymeth.2016.08.014.

[5] Guo, H., Wang, L., Chen, F. et al. “Scientific big data and Digital Earth”, Chin. Sci. Bull. (2014) 59: 5066–5073, https://doi.org/10.1007/s11434-014-0645-3.

[6] Ming-Hsiang Tsou, “Research challenges and opportunities in mapping social media and Big Data,” Cartography and Geographic Information Science, vol. 42, pp. 70–74, 2015, Taylor & Francis, https://doi.org/10.1080/15230406.2015.1059251.

[7] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, Samee Ullah Khan, “The rise of “big data” on cloud computing: Review and open research issues,” Information Systems, Volume 47, 2015, Pages 98–115, https://doi.org/10.1016/j.is.2014.07.006.

[8] A. Weiss, Computing in the Cloud, vol. 11, ACM Networker, 2007, 18—25.

[9] I. Foster, Y. Zhao, I. Raicu and S. Lu, “Cloud Computing and Grid Computing 360-Degree Compared,” In Grid

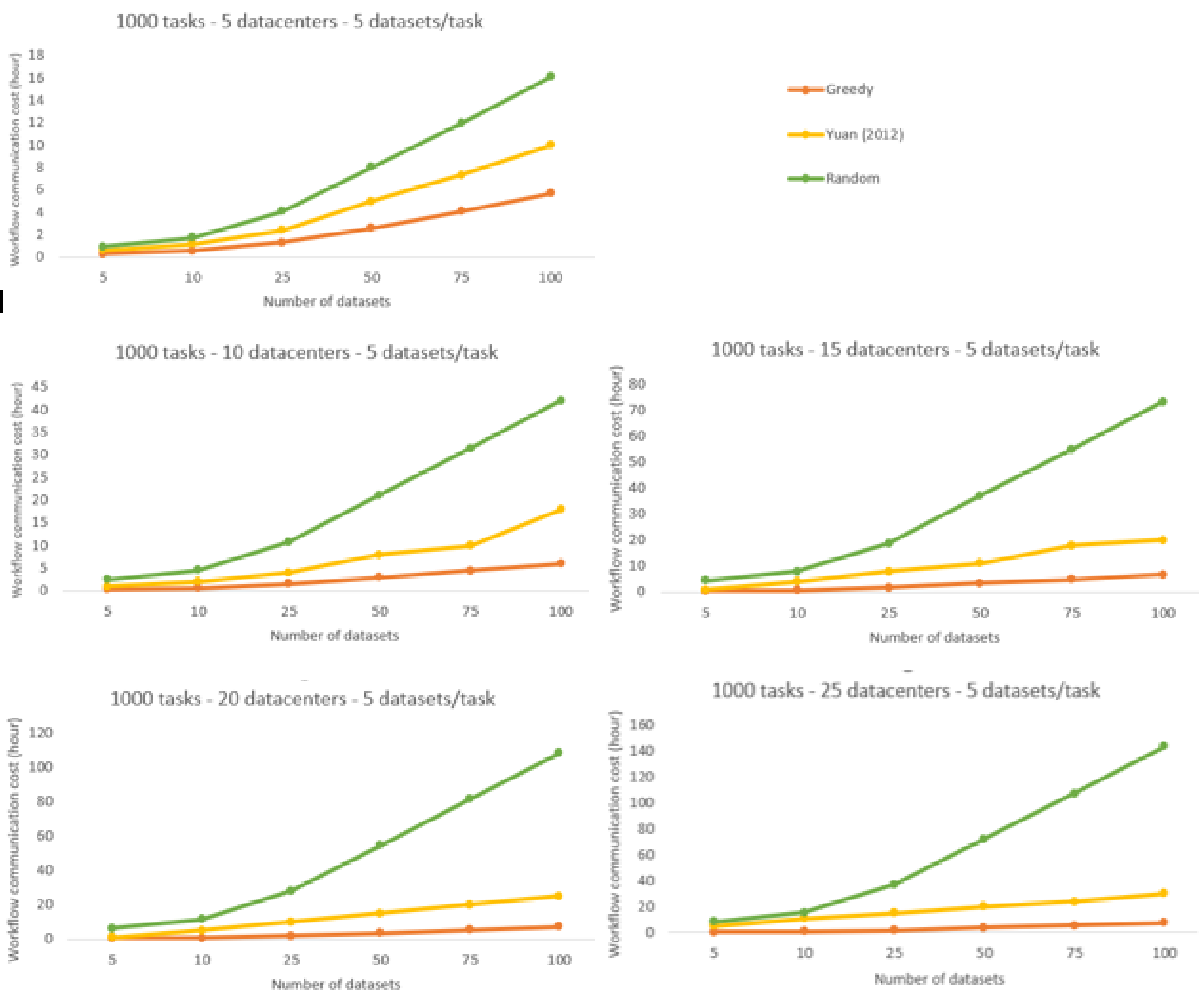

Fig. 2. Workflow communication cost among datacenters by varying the number of datasets while fixing the number of datacenters

Computing Environments Workshop, 2008. GCE'08, pp. 1–10, 2008.

[10] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. 2011. "Big data and cloud computing: current state and future opportunities." In Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11), Anastasia Ailamaki, Sihem Amer-Yahia, Jignesh Pate, Tore Risch, Pierre Senellart, and Julia Stoyanovich (Eds.). ACM, New York, NY, USA, 530-533, http://dx.doi.org/10.1145/1951365.1951432.

[11] R. Ranjan,"Streaming Big Data Processing in Datacenter Clouds," in IEEE Cloud Computing, vol. 1, no. 1, pp. 78-83, May 2014, http://dx.doi.org/10.1109/MCC.2014.22.

[12] Khan, Z., Anjum, A., Soomro, K. et al. "Towards cloud based big data analytics for smart future cities." J Cloud Comp 4, 2 (2015), http://dx.doi.org/10.1186/s13677-015-0026-8.

[13] Jorge L. Reyes-Ortiz, Luca Oneto, Davide Anguita, "Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf," Procedia Computer Science, Volume 53, 2015, Pages 121–130, https://doi.org/10.1016/j.procs.2015.07.286.

[14] Data Center Global Expansion Trend.

[15] Lu. Zhihui, Wu. Jie, Bao. Jie, Hung. Patrick, (2016). "OCReM: OpenStack-based cloud datacentre resource monitoring and management scheme," International Journal of High Performance Computing and Networking. 9. 31. 10.1504/IJHPCN.2016.074656.

[16] Agrawal D., El Abbadi A., Antony S., Das S. (2010). "Data Management Challenges in Cloud Computing Infrastructures," 1–10. https://doi.org/10.1007/978-3-642-12038-1_1.

[17] D. Yuan, Y. Yang, X. Liu, J. Chen, "A data placement strategy in scientific cloud workflows", Future Genera-

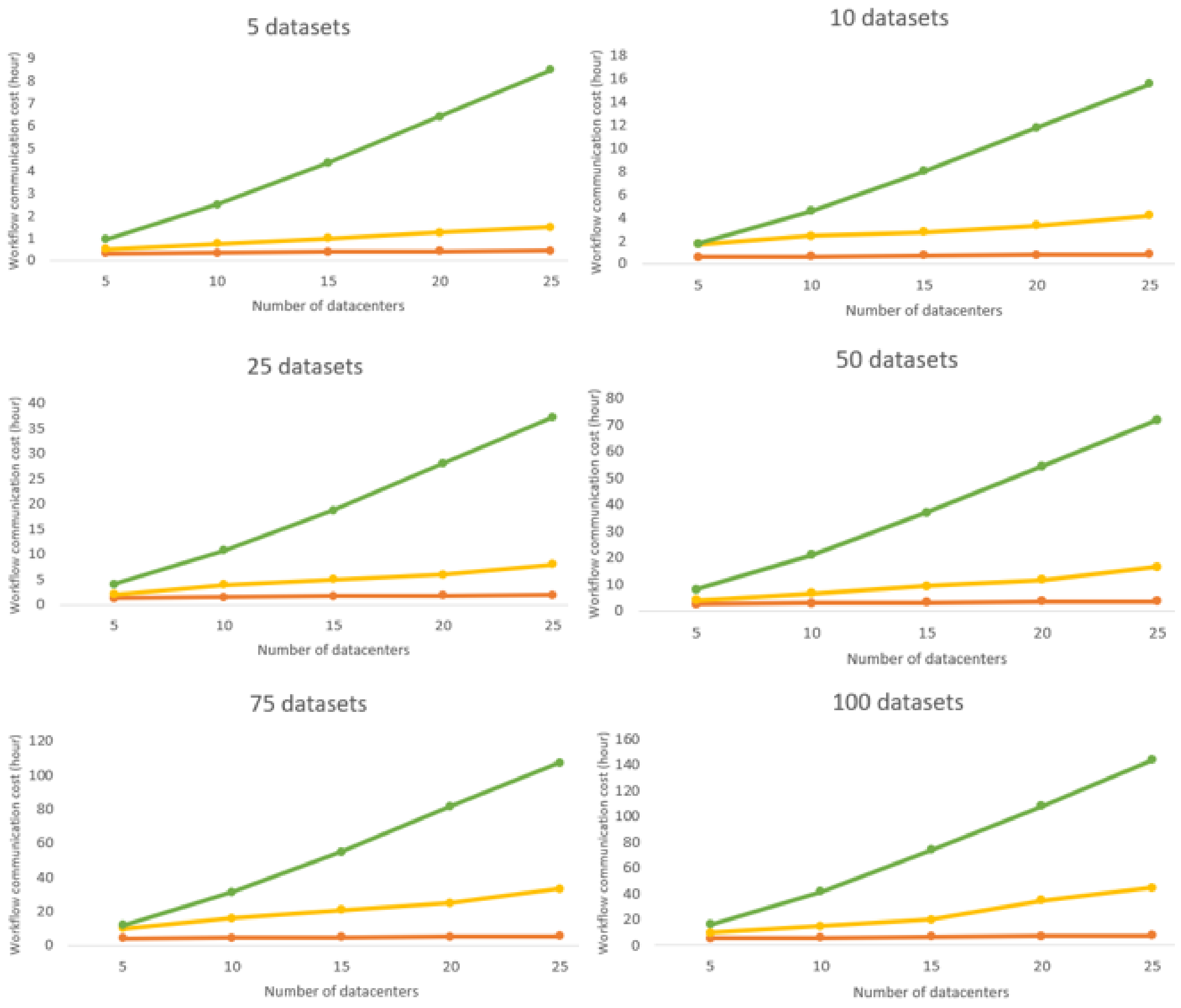

Fig. 3. Workflow communication cost among datacenters by varying the number of datacenters while fixing the number of datasets

tion Computer Systems, Volume 26, Issue 8, 2010, Pages 1200–1214, https://doi.org/10.1016/j.future.2010.02.004.

[18] Xu, Qiang & Xu, Zhengquan & Wang, Tao. (2015). A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing. International Journal of Intelligence Science. 05. 145-157, https://doi.org/10.4236/ijis.2015.53013.

[19] Z. Er-Dun, Q. Yong-Qiang, X. Xing-Xing and C. Yi, "A Data Placement Strategy Based on Genetic Algorithm for Scientific Workflows," 2012, Eighth International Conference on Computational Intelligence and Security, Guangzhou, 2012, pp. 146–149, https://doi.org/10.1109/CIS.2012.40.

[20] Q. Zhao, C. Xiong, X. Zhao, C. Yu and J. Xiao, "A Data Placement Strategy for Data-Intensive Scientific Workflows in Cloud,"2015, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, 2015, pp. 928–934, https://doi.org/10.1109/CCGrid.2015.72

[21] M. Ebrahimi, A. Mohan, A. Kashlev and S. Lu,"BDAP: A Big Data Placement Strategy for Cloud-Based Scientific Workflows," 2015 IEEE First International Conference on Big Data Computing Service and Applications, Redwood City, CA, 2015, pp. 105–114, https://doi.org/10.1109/BigDataService.2015.70

[22] R. N. Calheiros, R. Ranjan, A. B., C. De Rose, and R. Buyya. 2011. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software Prac. Experience 41, Issue 1 (Jan 2011), 23–50.

https://doi.org/10.1002/spe.995

[23] L. Bouhouch, M. Zbakh, C. Tadonki. Data Migration: Cloudsim Extension, The 3rd International Conference on Big Data Research (2019), pp. 177-181. https://doi.org/ 10.1145/3372454.3372472.