



HAL
open science

Cost-efficient big intermediate data placement in a collaborative cloud storage environment

Sonia Ikken, Eric Renault, Amine Barkat, Abdelkamel Tari, Tahar Kechadi

► To cite this version:

Sonia Ikken, Eric Renault, Amine Barkat, Abdelkamel Tari, Tahar Kechadi. Cost-efficient big intermediate data placement in a collaborative cloud storage environment. 2017 IEEE 19th International Conference on High Performance Computing and Communications, IEEE 15th International Conference on Smart City and IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Dec 2017, Bangkok, Thailand. pp.514-521, 10.1109/HPCC-SmartCity-DSS.2017.67 . hal-04390441

HAL Id: hal-04390441

<https://hal.science/hal-04390441v1>

Submitted on 13 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cost-Efficient Big Intermediate Data Placement in a Collaborative Cloud Storage Environment

Sonia Ikken^{*†}, Éric Renault^{*}, Amine Barkat^{‡†}, Abdelkamel Tari[†] and Tahar Kechadi[‡]

^{*}Telecom SudParis, Samovar-UMR 5157 CNRS, University of Paris-Saclay, France

[‡]UCD School of Computer Science and Informatics, Dublin, Ireland

[§]Politecnico Di Milano in Italy

[†]Faculty of Exact Sciences, University of Bejaia, 06000 Bejaia, Algeria

Email: {sonia.ikken, eric.renault}@telecom-sudparis.eu, amine.barkat@polimi.it, tarikamal59@gmail.com, tahar.kechadi@ucd.ie

Abstract—Collaborative cloud storage environment, which share resources of multiple geographically distributed datacenters owned by different providers enable scientific workflow from different locations to process large scale big intermediate data through the Internet. Distributed datacenters are federated and each member can collaborate with each other to efficiently share and process the intermediate data from distributed workflow instances. This paper focuses on the storage cost minimization of intermediate data placement in federated cloud datacenters. Through collaborative and federation mechanisms, we propose an exact federation data placement algorithm based on integer linear programming model (ILP) to assist multiple datacenters hosting intermediate data files generated from a scientific workflow. Under the constraints of the problem, the proposed algorithm finds an optimal intermediate data placement with a cost saving over the federated cloud datacenters, taking into account scientific user requirements, data dependency and size. Experimental results show the cost-efficiency of the proposed cloud storage federation algorithm.

Keywords—Collaborative Cloud Storage Environment, Big data Scheduling, Storage Cost Saving, Intermediate data Dependency

I. INTRODUCTION

Big data workflows have become an important paradigm since the introduction of scientific workflow and the need to formalize complex data-intensive scientific processes. One common characteristic of big data workflow applications is the existence of intermediate data during the execution of workflow instances. This leads to the generation of a massive amount of intermediate results as data dependencies need to be hosted and managed over a cloud infrastructure. Handling large intermediate data dependencies in cloud infrastructure is important for such operations and need a long time for execution since those dependencies need to process intermediate results from different storage locations. As some intermediate data are very large to be relocated efficiently, this operation must take into account the dependencies between intermediate data in selecting their locality. Furthermore, scientific users share important intermediate data dependencies for cooperation and reproduction of new intermediate results. This has enabled researchers to collaboratively work with other professionals or scientific users around the world and to handle and share intermediate data workflow enormously larger in size than before. By offering storage services in several geographically distributed datacenters, cloud infrastructure enables big data workflow applications to offer low latency access to scientific user data. However, the ever increasing volumes of scientific intermediate data address the need to interpret, move and store

them more efficiently to the most appropriate datacenter. One fundamental issue in dealing with such scales of scientific user intermediate data results for a workflow application is how to efficiently place them in a distributed cloud datacenter while ensuring the dependency and scalability of the placed data such that the total storage cost of the cloud provider is minimized. On another note, cloud storage providers offer geographically distributed datacenters providing several storage classes with different prices. They can collaborate by sharing their respective resources and dynamically adjust their hosting capacities in response to their data applications. An important problem faced by cloud users is how to exploit these storage classes to serve an application with data requirements at minimum cost. A federation of existing cloud storage services supports the scientific users with a unified and combined view of storage and data services across several providers and applications. Recently, several studies take advantage of pricing plan variety of different resources in a cloud storage federation, where the cost can be optimized by trading through negotiation a storage versus compute and network resources as well as cost optimization of data distribution across cloud providers [1], [2], [3], [4] (here, we are disregarding in profit improvements). None of these studies investigated the trade off between network and storage cost to optimize cost of data workflow placement across a federated cloud storage provider. Our study is motivated by these pioneer issues as none of them can simultaneously answer the aforementioned questions (i.e., placement and cost saving of data workflow in cloud storage federation).

Hence, addressing these questions makes the following main contributions: by exploiting cloud storage federation characteristics and data workflow requirements, we formalize and model the input and output parameters of the system cost model. We then propose a cost optimization problem in which the optimal cost of transferring, storing and requesting intermediate data is calculated. The exact intermediate data file dependency is assumed to be known a priori in order to focus on the data workflow placement problem itself. In this approach, we proposed an exact algorithm based on the integer linear programming model that takes into account the dependency requirements (valuable and unnecessary correlation) of these intermediate data for making decisions, and reallocates intermediate data requests with dependencies in a single datacenter to reduce the total storage cost.

This paper is organized as follows: Sec. II describes the system model based on the cloud storage federation scenario

with target assumptions. Sec. III derives an exact optimization approach for allocating intermediate data on federated Cloud storage. Sec. IV shows and discusses the simulation results obtained with compared scenarios. Related work is reviewed in Sec. V. Finally, Sec. VI concludes the paper and presents some future work.

II. SYSTEM MODEL

A. Scenario Assumptions

The scenario introduced in Fig. 1 illustrates a cooperation of cloud storage providers noted by P which involves federated datacenters D that are geographically distributed providing on-line mass storage to the scientific community collaborations to schedule a set of intermediate data noted by ID_i . i being a single file with its respective size noted $size_i$, d and d' indicate home datacenters where intermediate data are generated by a workflow instance and are temporarily stored.

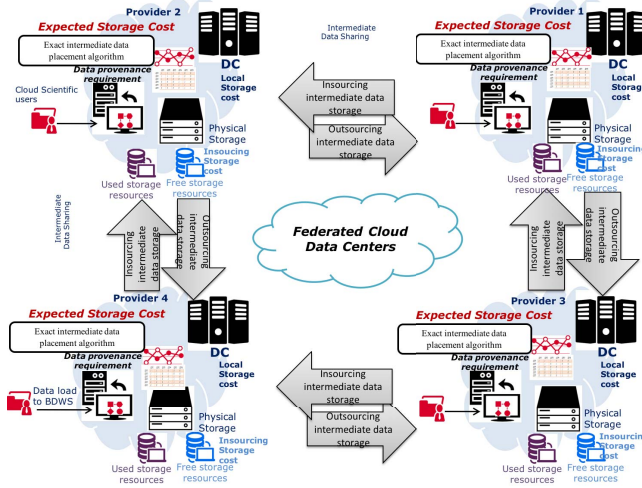


Fig. 1. Federated cloud datacenters scenario.

The set of federated datacenters D are aggregated and interconnected in the form of an inter-cloud. They use native peer-to-peer communication link to shift intermediate results from a busy disk entity to those with an available capacity, and efficiently use storage resources of the datacenters and balance intermediate data placement requirements. The bandwidth resource of each link is used to transfer the intermediate data files between federated datacenter noted k . The intermediate data placement decision on one or multiple locations of the cloud storage federation involves the use of datacenters converged in local and storage federation for data re-utilization internally or by the other collaborative users. After the workflow instances have generated intermediate data locally noted IDN_d on each datacenter, a memory temporary stores these intermediate data results before their scheduling and placement decision. The set of considered datacenters can be modeled as a matrix noted $DChome_d$ which combines a home datacenter in order to emphasize the origin of the generated data. For the sake of convenience, we here only consider storage and data transfer resources of D as computing resource provisioning is similar to that of the storage resource. Hence, to underscore the limited amount of available storage space on each federated datacenter k , each member in the federation knows the storage resource quotas that is capped at SCF_k offered by the other

federation members as well as their internal capacity that is capped at SCL_k . Denoted BCF_k , the data bandwidth shared between the home datacenter and the destination datacenters to transfer one unit of intermediate data file. Let $DBmax_k$ be the maximum data bandwidth quota (in GB per month) provided among federated datacenters.

Each provider that hosts federated datacenters cooperatively proposes a storage cost noted OSC_k , and induces a local storage cost noted LSC_d for the internally hosted intermediate data. A storage federation price varies according to busy storage or data bandwidth resources formulated by S_{busy} . Each federated datacenter provides a maximum cloud storage quota noted $QCmax_k$ to every users. Then, the busy quota is reduced from the maximum capacity of storage and bandwidth resources. However, cloud storage providers must rise to the variation of the price in real-time regarding to various factors such as data access demands, cloud storage market rates, and datacenter localization. To unlock these constraints, a pricing strategy [11] is adapted to determine the storage federation cost of the providers that varies according to their busy quotas in a way that when the busy quota is high the monetary cost goes up and when it is low the monetary cost goes down. This pricing mechanism allows them to dynamically set their insourcing/outsourcing storage or transfer prices by establishing the monetary cost in exchange of offering data storage space or data bandwidth or selling storage resources. Below, the mathematical formula determines the insourcing / outsourcing prices for the data placement demands:

$$S = \frac{SCmax_k - S_{dis}}{SCmax_k} * (S_{price} - ME_{price}) + ME_{price} \quad (1)$$

Basically, equation (1) considers the minimum effective price (ME_{price}) that is a reference monetary cost of the amount of data storage or data transfer that providers do not fall below in order to address economic issues. The affected cost (S_{price}) for the end-users is fixed and varies according to standard on-demand cloud storage pricing plan based on reservation contract or prepaid scheme [12]. Moreover, the maximum capacity of the storage federation $QCmax_k$ is given by the totality of the storage or transfer quota offered by each datacenter. A very important point to consider during the intermediate data placement on the storage federation is a data transfer in or from other federation members. It should be mentioned that in most cloud storage services, the monetary cost of data transfers is more expensive than the data storage itself, so this is an important cost factor that must be considered in our placement problem. Therefore, the transfer cost of data insourcing and data outsourcing are noted ITC_k and OTC_k respectively. The outsourcing and insourcing storage and transfer costs (OSC_k , LSC_d , ITC_k and OTC_k) are updated using equation (1), and depend on their available outsourcing versus local capacity and the minimum effective storage price of each generated intermediate data file.

B. Matrix Model for Intermediate data dependency

As stated earlier, each federated datacenter k receives intermediate data placement requests from multiple home datacenters. When new intermediate data files are generated by an instance workflow, correlations between each file pair are revealed. These correlations correspond to the inter-file dependencies generated from several workflow instances.

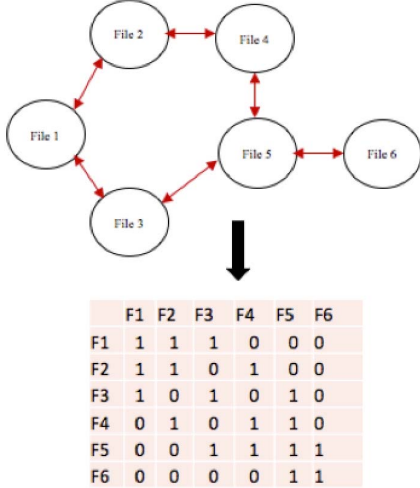


Fig. 2. Intermediate data dependency matrix

Then, let DEP be a binary integer matrix including a symmetric dependency value $Dep_{i,j}$ for each pair of files. Equation (2) exposes the $Dep_{i,j}$ values:

$$Dep_{i,j} = \begin{cases} 1 & \text{dependency between files } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Fig. 2 illustrates an example of intermediate data dependency and the corresponding matrix model. Each intermediate data file i has a dependency with itself and with some other file j , so if a set of intermediate data files ID_i that has a correlation with file j , this one is also correlated with the same set ID_i (symmetry). Therefore, input parameters of the data placement model are dependency values $Dep_{i,j}$ that are collected in matrix DEP .

However, various skewed of dependencies can be specified between the intermediate data during a freak execution of a workflow instance. Scientist users may encounter these errors¹ during the execution that causes unnecessary dependencies that need to be adjusted or re-run. Hence, some intermediate data dependencies are not valuable. In order to consider this situation, a float parameter λ_i^j is defined in the data placement model that denotes scientific users's tolerance of dependency files i and j :

$$\lambda_i^j = \begin{cases} 0 & \text{no tolerance to process independently } i \text{ and } j \\ 1 & \text{otherwise.} \end{cases} \quad (3)$$

The generated intermediate data dependencies with $\lambda_i^j = 1$, operate over a set of I/O requests between each other in the selected datacenter federation. Some operations can be involved in remote access requests: data input adjustment, re-processing or data re-utilization. For each I/O access, there is a cost noted $IOPC_{i,j}$. The values of the dependency matrix DEP is dynamically maintained for each set of generated files.

¹It can be an information about a replacement task, integrating a new input data that improves the reliability of the workflow instance execution.

Accordingly, between each pair of files (i, j) , value λ_i^j parameter is defined. The amount and the size of intermediate data dependencies feeds the expected storage cost when scheduling intermediate data ID_i on the federated locations. These later collaborate by sharing their respective storage resources and dynamically adjust their hosting capacities according to their intermediate data placement requirements.

III. EXACT ALGORITHM

In the following we resort to an exact approach called Exact Federation Big Data Workflow Placement algorithm (ExactFed_BDWP) which leads to a mathematical programming approach based on an Integer Linear Programming (ILP) model. The ExactFed_BDWP algorithm is solved using a branch-and-bound method describing a set of valid inequalities of the big data workflow placement problem cited above and the objective function to optimize under linear constraints. Some of these constraints are obtained according to a practical system in the cloud data placement considering service scenarios and storage capacities. Finding the optimal placement requires the computation of storage cost for each possible instance solution from each intermediate data placement request (input parameters) on a federated and local datacenters. The federated datacenters use a cost model and the ExactFed_BDWP algorithm according to the data dependency to schedule their intermediate data placement requests to the storage federation.

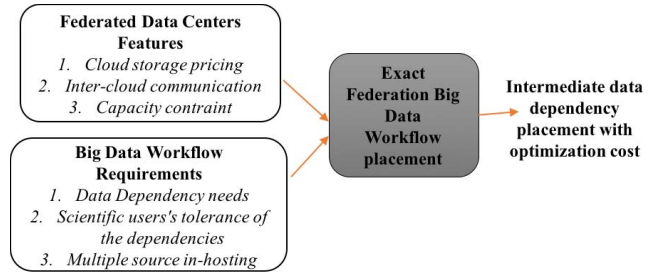


Fig. 3. Overview of the data storage cost federation approach: input/output parameters.

As shown in Fig. 3, the storage selection decision has to lead to the minimum data storage cost in compliance with the intermediate data dependency constraints (one datacenter hosting several intermediate data dependency files) and a maximum storage utilization for the federated datacenters. Hence, we discuss the cost model and the objective function that should be minimized considering the objective and the following procedures for the ExactFed_BDWP algorithm execution:

- From a set of input parameters $Dep_{i,j}$, each federated datacenter exposes its insourcing/outourcing storage cost based on the Equation (1).
- For each $Dep_{i,j}$ value, the cost of the newly calculated intermediate data placement solution is compared with the currently lowest cost placement in each federated datacenter. The ExactFed_BDWP algorithm terminates after the set of all relevant intermediate data placement solutions have been checked. The ExactFed_BDWP algorithm is performed under the requirements of intermediate data owner and capacity constraints from the selected home and federated datacenters.

TABLE I. STORAGE COST FEDERATION MODEL: PARAMETERS & DECISION VARIABLES.

| Parameters | Descriptions |
|-------------------|--|
| P | Set of cloud storage providers |
| D | Set of federated datacenters |
| d, d', k, k' | Designate home datacenters d and d' and federated datacenters k and k' |
| $Dep_{i,j}$ | Dependency coefficient value of intermediate data dependency as input to the model to designate affinity between files i and j |
| DEP | Dependency matrix including all dependency values of $Dep_{i,j}$ |
| $DChome_d$ | A home datacenter matrix which brings home datacenter d to generate intermediate data files |
| LSC_d | Local storage cost (dollar per GB) of cloud storage provider hosting datacenter d |
| OSC_k | Outsourcing storage cost (dollar per GB) of cloud storage provider hosting datacenter k |
| $size_i$ | Size of intermediate data file i |
| λ_i^j | Scientific users's tolerance of intermediate data dependencies of i and j which is a binary value, $\lambda_i^j = 0$ indicates scientific users have no tolerance to process independently i, j , and 1 otherwise. |
| i | Single intermediate data file |
| ID_i | Set of intermediate data files |
| IDN_d | Number of intermediate data generated and stored temporary in datacenter d |
| ITC_k | Insourcing transfer cost (dollar per GB) proposed by provider to transfer data from home datacenter k |
| OTC_k | Outsourcing transfer cost (dollar per GB) proposed by a provider to transfer data to another datacenter k' |
| $IOPC_{i,j}$ | Cost of I/O requests (dollar per operation) of intermediate data files i and j on federated datacenters |
| SCF_k | Storage space quotas offered and shared (GB per month) by each datacenter k in the storage federation |
| $SCmax_k$ | Maximum storage space quotas (GB per month) offered and shared by all the federated datacenters |
| SCL_d | Storage space quotas (GB per month) available by each home datacenter d |
| BCF_k | Data bandwidth (per data unit) quota offered and shared between home datacenter and destination datacenter to insource and outsource intermediate data storage |
| $DBmax_k$ | Maximum data bandwidth (GB) quotas provided among federated datacenters |
| Variables | Definitions |
| x_{id}^k | A binary variable, $x_{id}^k = 1$ if intermediate data file i is scheduled from datacenter storage d to outsourced datacenter k , and 0 otherwise. |
| $y_{ijdd'}^{kk'}$ | A binary variable, $y_{ijdd'}^{kk'} = x_{id}^k * x_{jd'}^{k'}$ |

- The ExactFed_BDWP algorithm keeps the intermediate data dependencies in a single datacenter while saving their storage, transfer and transaction costs. The intermediate data with dependency tolerance should be optimized according to the cost of I/O-demands requested among federated datacenters.
- The problem is solved on each workflow instance when a new intermediate data file is generated from a home datacenter. Such cost is assumed to change according to the federation features.

In the following we introduce the use of the integer bivalent variables 0-1 x_{id}^k that tells which datacenter d hosts intermediate data file i to be placed in federated datacenter k . A glossary of all the used notations and their descriptions in the proposed exact model is shown in Table 1. Using these notations, the global objective function is given by equation (4):

$$MinCost = \sum_{idk}^{d \neq k} x_{id}^k \cdot size_i \cdot (OSC_k + ITC_k + OTC_d) + \sum_{idk}^{d=k} x_{id}^k \cdot size_i \cdot LSC_k + \sum_{ijdd'k}^{d \neq k} y_{ijdd'}^{kk'} \cdot Dep_{ij} \cdot \lambda_i^j \cdot IOPC_{i,j} \quad (4)$$

The objective function for the optimal scheduling and placement of intermediate data files can be expressed as the minimization of the cost of transferring, storing data workflow in the federated datacenters, where $x_{id}^k = 1$ is used to indicate that intermediate data file i is transferred and placed in federated datacenter k and $x_{id}^k = 0$ otherwise. Ideally, the ExactFed_BDWP algorithm should minimize the I/O transaction cost also when intermediate data dependencies are scheduled separately to different datacenter, i.e. $y_{ijdd'}^{kk'} = 1$ and $y_{ijdd'}^{kk'} = 0$ otherwise. Objective function (4) is subject to several linear and integrity constraints expressed respectively by equations (5) to (16):

$$\sum_{ik} x_{id}^k = IDN_d \quad \forall d \in D; d \neq k \quad (5)$$

$$\sum_{dk} x_{id}^k = 1 \quad \forall i \in ID_i \quad (6)$$

$$x_{id}^k + x_{jd'}^{k'} = 2 \quad \forall i, j \in ID_i; \forall k, d, d' \in D \quad (7)$$

$$x_{id}^k + x_{jd'}^{k'} \leq 1 \quad \forall i, j \in ID_i; \forall k, d, d' \in D \quad (8)$$

$$x_{id}^k + x_{jd'}^{k'} - y_{ijdd'}^{kk'} \leq 1 \quad \forall i, j \in ID_i; i \neq j; \forall k, k', d, d' \in D \quad (9)$$

$$\sum_{kk'dd'} y_{ijdd'}^{kk'} \leq \sum_k x_{id}^k \quad \forall d \in D; \forall i, j \in ID_i \quad (10)$$

$$\sum_{id}^{k \neq d} x_{id}^k \cdot size_i \leq SCF_k \quad \forall k \in D \quad (11)$$

$$\sum_{id}^{k=d} x_{id}^k \cdot size_i \leq SCL_d \quad \forall k \in D \quad (12)$$

$$\sum_{id}^{k \neq d} x_{id}^k \cdot size_i \cdot BCF_k \leq DBmax_k \quad \forall k \in D \quad (13)$$

$$\sum_{id} x_{id}^k \cdot size_i \leq SCmax_k \quad \forall k \in D \quad (14)$$

$$\sum_d IDN_d = 1 \quad \forall i \in ID_i \quad (15)$$

$$Dep_{ij} = Dep_{ji} \quad \forall i, j \in ID_i; \forall Dep_{ij} \in DEP \quad (16)$$

Equation (5) expresses the placement constraint in federated datacenters which store the set of ID_i files. Equation (6) indicates that there is only one datacenter hosting file i . Equations (7) and (8) express dependency constraints that are derived from the values of dependency tolerance λ_i^j which indicates the splitting ($\lambda_i^j = 1$) or the merging ($\lambda_i^j = 0$) of the intermediate data files during their placement in the federation. Equations (9) and (10) define the relations between bivalent variables x_{id}^k , $x_{jd}^{k'}$ and $y_{ijdd}^{kk'}$. Equations (11) and (12) define the storage capacity constraints in each home and federated datacenter with a limited quota. The data bandwidth quota offered to transfer intermediate data files from or out of the federated datacenter is expressed by equation (13) and the aggregation of these quotas cannot exceed the provided maximum bandwidth ($DBmax_k$). The storage resources limitation of the federation is expressed by equation (14). Equation (15) indicates the uniqueness constraint of each generated file i from a single workflow instance. The relation of dependency between files i and j is symmetric and is expressed by equation (16).

IV. PERFORMANCE EVALUATION

A. Setting Parameters

In order to evaluate the proposed model and to show the influence of using federated cloud storage characteristics, a set of simulations have been performed with different input parameters. The evaluation model is performed under AMPL tools and the CPLEX solver² as an ILP optimization program to solve objective function (4). The assessment concerns the cost optimization of intermediate data dependency placement. To create a dynamic environment and unpredictable situations, between 3 and 18 geographically distributed datacenters have been randomly selected from three cloud providers. Among these datacenters, 8 are owned by Amazon S3 (AM)³, 4 by Google Cloud Storage (GO)⁴ and 6 by Microsoft Azure (MZ)⁵. Each datacenter is restricted by storage and bandwidth capacities ranging in [10GB, 1000GB] and in [1GB,10GB] respectively. Each outsourcing and insourcing demand is composed of a random inter-file dependency ranging in [50*50,1000*1000] organized in a matrix with 1 to 2 GB size per file. Intermediate data files are affected to their home datacenter in a binary matrix ($DChome_d$). The insourcing/outsourcing storage and transfer monetary costs are given by equation (1). The I/O request cost and the affected monetary cost (S_{price}) for the end-users including OSC_k , ITC_k and OTC_k are set according to the pricing plan for each provider. Table 2 summarizes these different price ranges. For an economic market purpose, the minimum effective price (ME_{price}) is set randomly and is higher or close to the affected cost for the scientific community ($ME_{price}=S_{price}*0,45$). The binary value of λ_i^j is set randomly for each new generated intermediate data file.

B. Compared scenarios & performance metric descriptions

Since previous studies on data placement and cost saving of intermediate data dependencies in cloud storage federation differ and are not sufficiently close to the placement problem (see Sec. V) that our approach deals with (the involvement

TABLE II. STORAGE PRICES OF THE THREE CLOUD STORAGE PROVIDERS.

| Cloud storage providers | I/O cost (\$/10000 operations) | Storage (\$/GB) | Data transfer IN (\$/GB) | Data transfer OUT (\$/GB) |
|-------------------------|--------------------------------|-----------------|--------------------------|---------------------------|
| Amazon S3 | [0.005-0.01] | [0.004-0.04] | [0] | [0-0.25] |
| Google Cloud Storage | [0-0.005] | [0.007-0.023] | [0] | [0.08-0.23] |
| Microsoft Azure | [0.015-0.0345] | [0.08-0.125] | [0] | [0-0.181] |

of intermediate data dependencies at the lowest cost in the federated placement), we resort to a comparison with two following strategies: no-federation aspect on the one hand and a capacity-based placement strategy used in default Hadoop implementation on the other hand. Datacenters in the non-federation scenario turn in an autonomous way and depend on their own storage space resources to place intermediate data files. To elaborate this scenario, a relaxation of the ILP was built and consists in eliminating constraints (5), (7), (8), (11), (12) and (14). The intermediate data placement is scheduled entirely in each home datacenter thanks to the unlimited storage capacity (no loss). The outsourcing/ insourcing storage costs are obviously not integrated to solve the non-federation scenario considering just local dependencies. In a capacity-based scenario, the federated datacenters (nodes in the cluster Hadoop) randomly select the outsourcing storage to schedule the intermediate data files to the federation members only when their own resources are not available (nodes capabilities). Here, the selection is done arbitrarily to outsource intermediate data files without considering the dependencies (constraints 7 and 8).

We applied the following metrics to analyze the performance of the ExactFed_BDWP algorithm with the compared scenarios: (i) Total storage cost: this metric is defined by the objective function computed by equation 4 that measures the cost of transferring, storing and requesting intermediate data files to fulfill the evolving big data workflow requirements. This corresponds to the sum of all defined costs. (ii) Federation utilization: this metric shows the fairness of the intermediate data distribution on a selected datacenter in the federation. It is defined as the ratio between the amount of storage space used by intermediate data placement (both local and federation members) and the maximum amount of storage space for all intermediate data files placement. (iii) Convergence time: this metric measures the execution time of the ExactFed_BDWP algorithm in order to assess how fast the algorithm finds a solution to fulfill the intermediate data dependency placement.

C. Simulation results

The total storage cost evaluation of all algorithms through simulations is presented in Fig. 4 and 5. The results that show the fair use of the federation are summarized in Fig. 6, 7 and 8 Finally, Fig. 9 and 10 depict the execution time of the ExactFed_BDWP algorithm.

Fig. 4 depicts the results of minimizing the total storage cost of the ExactFed_BDWP algorithm with no-federation and

²<http://ampl.com/products/solvers/solvers-we-sell/cplex/>

³<https://aws.amazon.com/fr/s3/pricing/>

⁴<https://cloud.google.com/storage/pricing>

⁵<https://azure.microsoft.com/fr-fr/pricing/details/storage/>

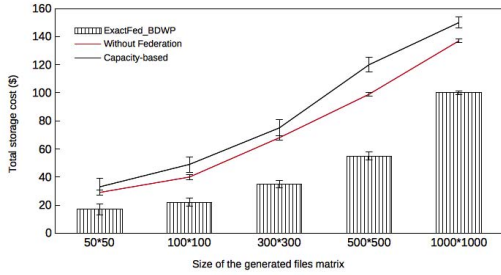


Fig. 4. The results of the optimal total storage cost regarding to dependency matrix size variation (DEP).

capacity-based scenarios while the number of datacenter is set to 9. The simulation results (100\$, 137\$ and 150\$ respectively for the extreme case) correspond to the aggregation of those obtained for each federated datacenter that participate and receive outsourcing and insourcing storage demands of intermediate data placement with varying amounts of files (DEP matrix size variation). The general observation is that ExactFed_BDWP algorithm testifies significant cost savings as compared to the benchmark scenarios. As expected, results show that the ExactFed_BDWP algorithm outperforms the compared strategies with 27.00 % in average total storage cost saving as regarding to the non-federation scenario and 33.33 % as compared to the capacity-based scenario while matrix size reached 1000*1000 which corresponds to 1000 files of 1 GB. Fig. 5 extends the cost saving evaluation for the ExactFed_BDWP algorithm by reporting performance as a function of dependency file pairs size ($Dep_{i,j}$) while the number of datacenters is set to 9.

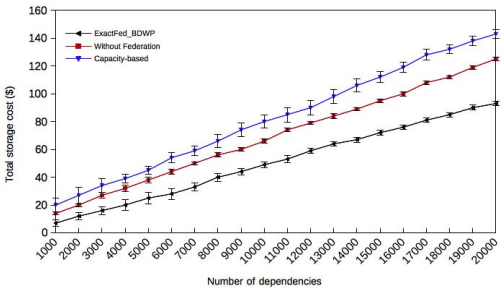


Fig. 5. Results of the optimal total storage cost regarding to the number of dependency file pairs ($Dep_{i,j}$).

With the increase of the size of the matrix and dependency files, the amount of stored intermediate data obviously increases in ExactFed_BDWP, non-federation and capacity-based algorithms (100\$, 137\$ and 150\$ respectively), and this influences much more the two scenario comparisons. Admittedly, the cost of insourcing/outsourcing file transfers are not included in the non-federation scenario, whereas the margin between its prices and those offered dynamically in the federation approach impacts on the total cost since the affected monetary costs are not negotiated and consider only local dependencies and take a fixed cost of home datacenter regardless to the price ranges. Similarly, for the capacity-based scenario that exhibits the highest average cost in the both Fig. 4 and 5, their costs corresponding to insourcing/outsourcing transfer,

storage and I/O cost substantially contrast with the total cost saving. It does not optimize the movement of intermediate data since it places them randomly to the different datacenters until the capacity is full without taking its dependencies in to consideration. Although the number of dependencies increases, the insourcing/outsourcing cost of intermediate data dependencies is minimized in the federation and this influences the total storage cost saving. In addition, the I/O request cost is minimized when there is a dependency tolerance.

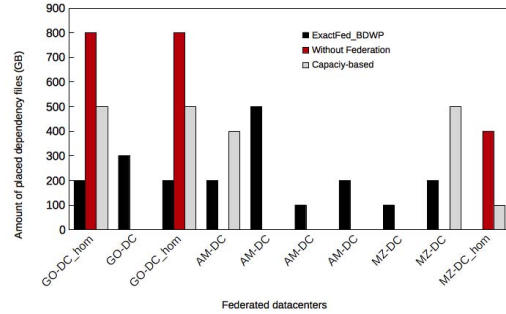


Fig. 6. Intermediate data distribution results for 6 federated datacenters.

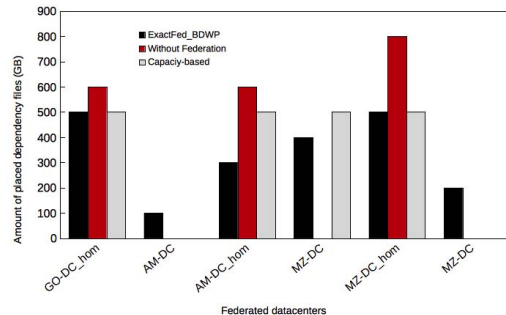


Fig. 7. Intermediate data distribution results for 10 federated datacenters.

The very important point in the federation is the placement balancing and fairness between federation members D for the intermediate data distribution. To achieve this, the amount of dependency files is set to 2000 GB (1000*1000 matrix size of 2GB per file) and are placed in federated datacenters and by both comparison scenarios by randomly setting the home datacenters from the providers P . The results are summarized in Fig. 6, 7 and 8 for a number of datacenters set to 6, 10 and 18 respectively those being selected for file placement decisions.

Figures show that the intermediate data placement by the federated datacenters are the most balanced overall from heterogeneous capacities and prices (with the exception of the unlimited capacities for no-federation scenario). The ExactFed_BDWP algorithm involves 6/6, 9/10 and 15/18 of the selected federated datacenters to schedule dependency files as compared to 4/6, 5/10 and 5/18 for the capacity-based scenario and 3/6, 3/10, 5/18 only for the scenario without federation in Fig. 6, 7 and 8 respectively. In fact, the storage federation contributes greatly and maintains the data distribution balancing among the members whatever the cloud storage provider participating (AM, GO and MZ). Moreover, the

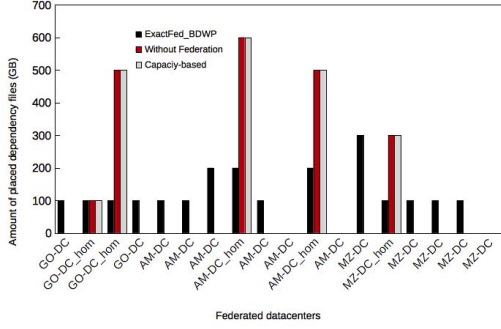


Fig. 8. Intermediate data distribution results for 18 federated datacenters.

negotiated attractive prices influence on the placement decision as each provider tries to offer dynamic pricing that balances the placement decision based on their capacity (equation (1)), thus maintaining a collaboration among federated datacenters by fairly placing the intermediate data dependency distribution in the cloud federation, thus reducing the charge for the scientific community. By contrast, as long as the no-federation scenario participates individually, the placement and storage cost is not optimal due to the unlimited capacities (the entire amount of data stored is linear with expected cost) and fixed prices, and consequently the amount of intermediate data dependencies are distributed only in the home datacenters.

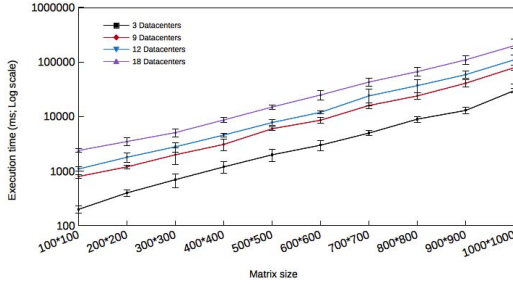


Fig. 9. Execution time of the ExactFed_BDWP algorithm by varying the number of federated datacenters.

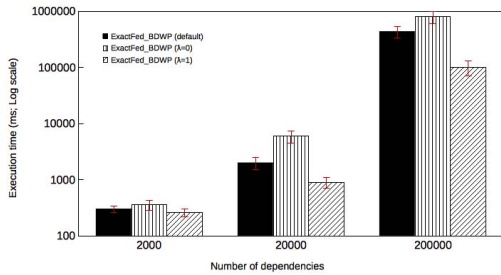


Fig. 10. Execution time of the ExactFed_BDWP algorithm with the different dependency parameters values λ_i^j while the number of datacenters is fixed to 9.

Fig. 9 and 10 pursue the analysis for the ExactFed_BDWP algorithm time execution by reporting performance respectively as a function of the matrix size (1000 GB of intermediate

data files) vs. the federation size that ranges in 3 to 18 datacenters, and dependency sizes (obtained from the aggregation of dependency file pairs $Dep_{i,j}$) with values 2000, 20000 and 200000 vs. the dependency values (the number of datacenters is fixed to 9 for the results in Fig. 10). As the problem is NP-hard (limitations of the branch-and-bound method), the execution time of the ExactFed_BDWP algorithm grows like the matrix size, especially when the number of datacenters is beyond 9 as reported in Fig. 9 (from 2 seconds to 4 minutes for all simulated instances). Fig. 10 illustrates the evaluation results of the influence of dependency constraints (expressed in equation (7) and equation (8)) on the performance of the ExactFed_BDWP algorithm. This corresponds to the cases: random default constraints (one could not force different the dependency constraints), splitting constraint (dependency files must be scheduled to different destination datacenters), and strong constraints (dependency files must to be scheduled to the same destination datacenters) while the number of datacenters in the federation is set to 9.

For small number of dependencies (2000) the ExactFed_BDWP algorithm exhibits close performance irrespective of the constraints. However, for the larger number of dependencies (20000 and 200000), more important differences appear with splitting, default and strong dependency constraints standing out as an intermediate data dependency placement problem whose resolutions are one minute to fifteen minutes. With the splitting constraint, the dependency file placement problem is solved faster as the space of feasible solutions is small (no inter-file dependencies to be considered in the resolution space). By contrast, in the case of strong dependency constraint the placement problem becomes harder to solve as there is a huge inter-file dependency to be considered in the resolution space while satisfying all storage requirements.

V. RELATED WORK

Data storage optimization has received a lot of attention in data workflow systems, and some previous works used the cost model to provide certain features from cloud storage scenario. Works presented in [5], [6], [7] are closer to our focusing problem. Authors in [5] elaborated a cost-effective strategy for storing intermediate data workflow in a single cloud storage provider using the Amazon-based fixed pricing. The proposed model focuses on running a scientific workflow system in a cloud and automatically deciding whether intermediate dataset should be stored or deleted in the cloud storage provider considering user's tolerance of computation delays. Since our allocation strategy does not remove intermediate data sets before the end of the workflow system processing, as shown in the introduction, all intermediate data sets are stored in multi-datacenters involving a cloud storage federation with a cost storage optimization. This differs considerably from [5] which cannot be compared to our model. Authors in [6] present a matrix-based k-means clustering strategy for data placement in scientific cloud workflow. The authors stress the movement of large volumes of data that can automatically be allocated among datacenters based on the data dependencies. The optimization is done only at the data movement level and the authors did not defined a storage cost optimization during the intermediate data placement which differs from our approach. In addition, our approach takes into account the type of dependency in order to further optimize the data movement and storage cost. Authors in [7] are closer to the

work in [6]. The authors present a data placement strategy based on data dependency clustering for scientific workflow in a heterogeneous cloud. The storage cost in this work is not considered and authors focus on the cost of the data redistribution only.

A cloud storage cost-based selection decision is presented in [8], [9], [10]. Authors in [8] propose an optimization problem for selecting the best storage services and they take into consideration application requirements and user priorities. A total storage cost is solved and workload requirements are not addressed in this work. In [9], the authors identify an adaptive-cost optimization system for multi-cloud storage to decrease the storage cost, a compression and placement algorithm are used to reduce data cost in cloud storage. However, the data correlation is not considered on the optimization cost model. Moreover, our approach highlights the trade-off between cost and data dependencies that is not the goal in [9]. The authors in [9] reduce the storage cost by compressing data and choose the lowest provider price. Intuitively, their cost model is a little more economical compared to our (by reducing the data size), but the fact of compressing data does not allow to deal with the dependencies in each storage providers which is a disadvantage for our optimization case. But the use of the federation storage price can balance considerably the costs of our optimization model in our favor in some input parameters. Authors in [10] propose a storage-cost optimization based on linear programming model using multiple public cloud storage providers. This work differs from our model since the type of data and collaboration-aspect-based cloud providers are not envisaged. Moreover, objectives of their optimization model are not exactly similar to ours, one of which is latency and the exclusion of communication cost between providers.

Since our optimization goal differs considerably from that of [5], [6], [7], [8], [9], [10], [10] it is not advisable to compare performance to these approaches.

VI. CONCLUSION

The present work introduced intermediate data placement cost saving solution through a collaborative cloud storage environment. An exact federation algorithm (ExactFed_BDWP) based on an integer linear programming (ILP) model and the branch-and-bound method has been proposed to solve the problem of the inter-file placement (symmetric dependencies) that takes into account the storage federation characteristics. The ExactFed_BDWP schedules and places fairly intermediate data files taking into account their dependency requirements, size and cost saving over distributed datacenter. A binary symmetric matrix is defined to represent the dependency for each pair of generated file in the same matrix, and home datacenter hosting matrix are used to outsource intermediate data storage to the federation. The ExactFed_BDWP algorithm was tested and evaluated by the way of simulations on a set of data files generated randomly with two different scenarios. An effective and optimal solution in terms of total storage cost saving was shown by the ExactFed_BDWP algorithm against the other scenarios. The execution time of the ExactFed_BDWP algorithm increases with the size of the dependency matrix, and the number of datacenters involved in the federation. However, for a realistic numbers of federated datacenters, the ExactFed_BDWP algorithm remains fast and achieves optimal scheduling and placement of intermediate data dependency.

The convergence time of ExactFed_BDWP algorithm also

matters in terms of swift response to additional data placement since some workflow applications require the placement of large data sets corresponding to a very complex dependency between a set of data files (asymmetric). This can thus put very stringent requirements on extended data placement, thus we plan in future work to consider a dependency types between a set of intermediate data.

REFERENCES

- [1] Rebai, S., Hadji, M., Zeghlache, D. (2015, January). Improving profit through cloud federation. In *Consumer Communications and Networking Conference (CCNC)*, 2015 12th Annual IEEE (pp. 732-739). IEEE.
- [2] Moreno-Vozmediano, R., Montero, R. S., Llorente, I. M. (2012). IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45(12), 65-72.
- [3] Vernik, G., Shulman-Peleg, A., Dippl, S., Formisano, C., Jaeger, M. C., Kolodner, E. K., Villari, M. (2013, June). Data on-boarding in federated storage clouds. In *Cloud Computing (CLOUD)*, 2013 IEEE Sixth International Conference on (pp. 244-251). IEEE.
- [4] Bermbach, D., Kurze, T., Tai, S. (2013, March). Cloud federation: Effects of federated compute resources on quality of service and cost. In *Cloud Engineering (IC2E)*, 2013 IEEE International Conference on (pp. 31-37). IEEE.
- [5] Yuan, D., Yang, Y., Liu, X., Zhang, G., Chen, J. (2012). A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience*, 24(9), 956-976.
- [6] Yuan, D., Yang, Y., Liu, X., Chen, J. (2010). A data placement strategy in scientific cloud workflows. *Future Generation Computer Systems*, 26(8), 1200-1214.
- [7] Zhao, Q., Xiong, C., Zhao, X., Yu, C., Xiao, J. (2015, May). A data placement strategy for data-intensive scientific workflows in cloud. In *Cluster, Cloud and Grid Computing (CCGrid)*, 2015 15th IEEE/ACM International Symposium on (pp. 928-934). IEEE.
- [8] Ruiz-Alvarez, A., Humphrey, M. (2012, May). A model and decision procedure for data storage in cloud computing. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)* (pp. 572-579). IEEE Computer Society.
- [9] Agarwala, S., Jadav, D., Bathen, L. A. (2011, July). iCostale: adaptive cost optimization for storage clouds. In *Cloud Computing (CLOUD)*, 2011 IEEE International Conference on (pp. 436-443). IEEE.
- [10] Negru, C., Pop, F., Cristea, V. (2014). Cost optimization for data storage in public clouds: a user perspective. In *Proceedings of 13th International Conference on Informatics in Economy*.
- [11] Toosi, A. N., Calheiros, R. N., Thulasiram, R. K., Buyya, R. (2011, September). Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment. In *High Performance Computing and Communications (HPCC)*, 2011 IEEE 13th International Conference on (pp. 279-287). IEEE.
- [12] Mazrekaj, A., Shabani, I., Sejdiu, B. (2016). Pricing Schemes in Cloud Computing: An Overview. *International Journal of Advanced Computer Science and Applications*, 7(2), 80-86.