An Adaptive Data Placement Strategy in scientific workflows over Cloud Computing Environments

Heewon Kim, Yoonhee Kim*
Dept. of Computer Science
Sookmyung Women's University
Seoul, Korea
{gmldnjs0610, yulan}@sm.ac.kr

Abstract— **Data of scientific workflow applications are tend to be distributed over many data centers to be effectively stored, retrieved, and transferred among them. The result of an experiment with those data shows diverse execution performance depending on the placement of input and intermediate data which are generated during application execution. However, initial data placement strategy would not be the best plan for long running experiments because of the dynamic change of resource condition time to time. We propose an adaptive data placement strategy considering dynamic resource change for efficient data-intensive applications. The strategy consists of two stages that group the datasets in data centers during the build-time stage and dynamically clusters every time newly generated datasets repeatedly to the most appropriate data centers during runtime stage, which is based on task dependency, intense degree of data usage, and just-in-time resource availability. Just-in-time data placement coming with task execution is more efficient than the one with initialization stage of experiments in the aspect of resource utilization. Experiments show that data movement can be effectively reduced while the workflow is running**

Index Terms— **Data-intensive application, Data-locality, Cloud computing, Scientific workflow.**

I. Introduction

Data-intensive scientific cloud workflow applications need not only huge amounts of storage but also high-performance computing resources [1]. Many applications in fields such as astronomy [2] and computational fluid dynamics [3] analyze large input data, process intermediate data generated during the execution, and produce large-scale data as a final result. In Scientific workflow, large amounts of application data tend to be stored in multiple distributed data centers. The performance of experiments depend on the placement of data while executing tasks, which are elements for an a^{2}pplication. Thus, it is necessary to schedule appropriate placement where data is stored from initialization, intermediate, and final stages. In addition, Intermediate data generated during execution requires toe be accessed between different data centers, that is, requires data transmission. Therefore, data placement strategy is important for efficient experiments of data-intensive applications.

Cloud computing technologies uses new methods to develop scientific workflow systems [4]. Because cloud resource dynamically change, cloud computing can be utilized in scientific workflow system. The recent data placement strategy research considered data dependency [4], data size [5], and bandwidth [6] in the build-time stage. However, only data placement strategy at initial/build-time stage would not be an optimal plan for long term experiments as the dynamic change resource condition. For a long-run workflow application, it is also necessary to consider intense degree of data usage or just-in-time resource availability during the runtime stage.

In this paper, we propose an adaptive data placement strategy considering characteristics a data-intensive application, at the same time, just-in-time resource planning depending on the status change of resource availability. This strategy groups using BEA(Bond Energy Algorithm) [9] algorithm datasets in data centers by considering data dependency during build-time stage and clusters datasets repeatedly considering intense degree of data usage and just-in-time resource availability during runtime stage. By placing data with their dependencies and just-in-time resource availability just before executing data-intensive tasks, which is a part of an application, our strategy attempts to minimize execution time of application.

The rest of this paper is organized as follows: Section 2 presents the related work. Section 3 introduces the adaptive data placement strategy. Section 4 describes our experiment for the strategy described in Section 3. Finally, Section 5 presents our conclusions and future work.

II. RELATED WORK

In data-intensive workflow applications, data placement is an important issue. A lot of research has been done on data placement in existing distributed computing systems. Dong Yuan [4] proposed a data placement strategy using a matrix-based k-means clustering strategy in which data is placed according to dependency. Qing Zhao [5] is a study on data size, and proposed an offline strategy and an online strategy for data layout considering fixed replacement datasets. Qiang Xu [7]

* Corresponding author

places data using a genetic algorithm and minimizes data scheduling between data centers.

The research of the above papers can effectively reduce the movement of data in the cloud workflow application, but did not consider the state of the changed resources in the middle of execution. In this paper, we propose a strategy to re-place data according to the state of resources when the available resources change during execution.

III. STRATEGIES FOR DATA PLACEMENT

In this section, we propose an adaptive data placement strategy that takes into account the dynamically changing state of resources during the experiment based on the paper [4] that considers the dependency between data when placing data in data centers.

A. Basic strategies for data placement

The cloud system [8] is composed of multiple data centers, and cloud workflow applications are very complex and can require a large number of datasets to perform a single task, and many tasks require a single dataset, and some datasets can be used together in many tasks. Moving data between data centers can take the amount of time when running these applications. To reduce the amount of time it takes to a data to data centers before running the application, use a statically applied algorithm [4] rather than randomly placing the data. This algorithm places the dependency between data in the experimental planning stage in order to reduce the movement of data between the data centers.

In this paper, data replacement is performed considering the status of dynamically changing resources during execution considering characteristics of cloud workflow application that status of data continuously changes due to the intermediate data generated during execution.

B. Data grouping and replacement based on dependency

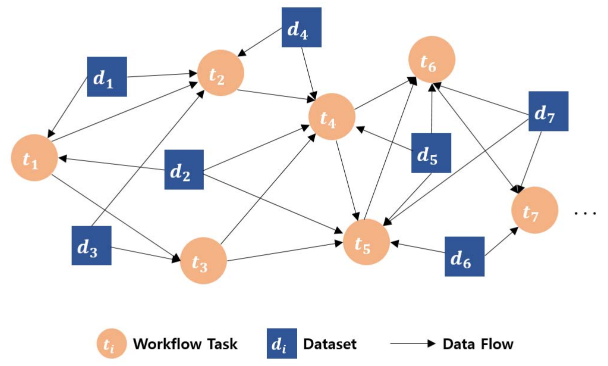


Fig. 1. Example of workflow

Figure 1 shows an example of a simple workflow between datasets and workflow tasks. In the Fig. 1, the circles marked as t_1, t_2 and etc. represent the tasks, and the squares marked as d_1, d_2 and etc. represent the datasets. The arrows indicate the data flow. The arrows indicate the data flow. For example, the data flow from dataset d_4 to task t_2 and task t_4 means that dataset d_4 is used at task t_2 and task t_4. The data flow from task t_4 to t_5 and t_6 means that the dataset generated by t_4 is used at both t_5 and t_6. As can be seen in Fig. 1, tasks such as t_2 and t_3 require two or more input datasets, and datasets such as d_2 are required for two or more tasks. Therefore, in a workflow system, there is always a complicated dependency between data. If a dataset is placed in consideration of this, data transmission can be effectively reduced.

$$\begin{array}{l} d_1 = \{t_1, t_2\} \\ d_2 = \{t_1, t_4, t_5\} \\ d_3 = \{t_2, t_3\} \\ d_4 = \{t_2, t_4\} \\ d_5 = \{t_4, t_5, t_6\} \\ d_6 = \{t_5, t_7\} \\ d_7 = \{t_6, t_7\} \end{array} \xrightarrow{\text{dependecy}} DM = \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline d_1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ d_2 & 1 & 3 & 0 & 1 & 2 & 1 & 1 \\ d_3 & 1 & 0 & 2 & 1 & 0 & 0 & 0 \\ d_4 & 1 & 1 & 1 & 2 & 1 & 0 & 0 \\ d_5 & 0 & 2 & 0 & 1 & 3 & 1 & 2 \\ d_6 & 0 & 1 & 0 & 0 & 1 & 2 & 2 \\ d_7 & 0 & 1 & 0 & 0 & 2 & 2 & 3 \end{array}$$

Fig. 2. Build up dependency matrix of Fig. 1

$$DM = \begin{array}{c|ccccccc} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \hline d_1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ d_2 & 1 & 3 & 0 & 1 & 2 & 1 & 1 \\ d_3 & 1 & 0 & 2 & 1 & 0 & 0 & 0 \\ d_4 & 1 & 1 & 1 & 2 & 1 & 0 & 0 \\ d_5 & 0 & 2 & 0 & 1 & 3 & 1 & 2 \\ d_6 & 0 & 1 & 0 & 0 & 1 & 2 & 2 \\ d_7 & 0 & 1 & 0 & 0 & 2 & 2 & 3 \end{array} \xrightarrow{\text{BEA}} CM = \begin{array}{c|ccccccc} & d_3 & d_1 & d_4 & d_2 & d_5 & d_7 & d_6 \\ \hline d_3 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ d_1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ d_4 & 1 & 1 & 2 & 1 & 1 & 0 & 0 \\ d_2 & 0 & 1 & 1 & 3 & 2 & 1 & 1 \\ d_5 & 0 & 0 & 1 & 2 & 3 & 2 & 1 \\ d_7 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\ d_6 & 0 & 0 & 0 & 1 & 1 & 2 & 2 \end{array}$$

Fig. 3. BEA transformation of dependency matrix of Fig. 1.

To do this, we calculated dependencies between the datasets and workflow tasks shown in Fig. 1, and created a dependency matrix DM. Dependency represents the number of common tasks of datasets in Figure 2[4]. Figure 3 shows the DM (Dependency Matrix) generated by calculating the dependency of Fig. 1 based on the algorithm in the paper [4]. DM is a $n \times n$ symmetric matrix, and n means the number of dataset. The value of the matrix is a calculation of the dependencies between the datasets and the workflow tasks in Fig. 2. Next, we use the BEA (Bond Energy Algorithm) [9] to transform the dependency matrix DM. BEA [9] is a permutation algorithm that can group similar items into matrices by replacing rows and columns. BEA creates a clustered dependency matrix CM, using the dependency matrix DM as input value. CM is a grouped matrix of similar values (small values with small values and large values with large values). All datasets are grouped according to clustered results and placed in the data center.

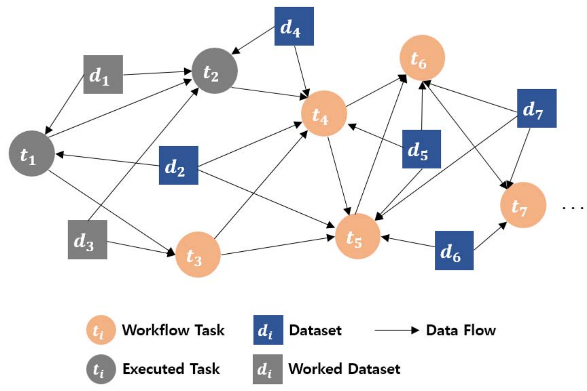


Fig. 4. Example of workflow during task execution

$$DM = \begin{array}{c|ccccc} & d_2 & d_4 & d_5 & d_6 & d_7 \\ \hline d_2 & 2 & 1 & 2 & 1 & 1 \\ d_4 & 1 & 1 & 1 & 0 & 0 \\ d_5 & 2 & 1 & 3 & 1 & 2 \\ d_6 & 1 & 0 & 1 & 2 & 2 \\ d_7 & 1 & 0 & 2 & 2 & 3 \end{array} \xrightarrow{BEA} CM = \begin{array}{c|ccccc} & d_2 & d_5 & d_7 & d_6 & d_4 \\ \hline d_2 & 2 & 2 & 1 & 1 & 1 \\ d_5 & 2 & 3 & 2 & 1 & 1 \\ d_7 & 1 & 2 & 3 & 2 & 0 \\ d_6 & 1 & 1 & 2 & 2 & 0 \\ d_4 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Fig. 5. BEA transformation of dependency matrix of Fig. 4

In this paper, we assume that when the tasks are executed to some extent, the task that was previously assigned to the resource is finished and the resource can execute the new task and the state of the resource changes unlike the first. Figure 4 shows the state changes between datasets and workflow tasks in data-intensive stage during task execution based on the paper [4]. In this case, data-intensive stages are determined by the pattern knowledge of previous executions. In the Fig. 4, colored circles and squares indicate that execution is terminated, and represent tasks and datasets that are no longer needed at run time. For example, datasets d2 and task t3 are needed for t5 to be executed later, but for d1 and t1, they are no longer needed during t4. Figure 5 shows the dependency matrix DM after again calculating dependencies between datasets and workflow tasks according to Fig. 4. Next, CM is generated using BEA [9] with DM as an input value. Based on the clustering result of the CM, it is re-grouped and placed in the data center considering the status of the changed resource. Looking at the two matrices DM and CM in Fig. 3 and Fig. 5, both before and during the execution of a task, we can see that the dependency between datasets and workflow tasks has changed and the state of the clustered dataset has changed as well.

In the next section, unlike the situation where new resources are created during the execution of the task, the strategy that replaces the data during the execution of the task instead of executing the data as it was before considering the dynamically changing state of the resource is called workflow execution time can be greatly reduced. Considering the state of the resource, a strategy that replaces it during the execution of the task shows that the workflow execution time can be greatly reduced.

IV. Experiments

In this section, we compared performance differences before and after data replacement to prove that they are efficient when replacing data, taking into account the state of resources dynamically changing during the execution of tasks mentioned in the previous section. We experimented and analyzed results. After describing the characteristics of the target application and explaining the experiment environment, finally, the experiment is explained and the results are analyzed.

A. Data-Intensive Scientific Workflow

In order to confirm the superiority of the proposed strategy, Montage GALFA [10] in the astronomical field with workflow characteristics was experimented. Montage is an Astronomical Image Mosaic Engine that generates composite FITS (Flexible Image Transport System) mosaics using multiple astronomical images. The application characteristics of Montage are memory-intensive and depend on execution depending on the memory of the resource. The Montage GALFA application uses a large amount of input data to generate a large number of intermediate data and one output data.

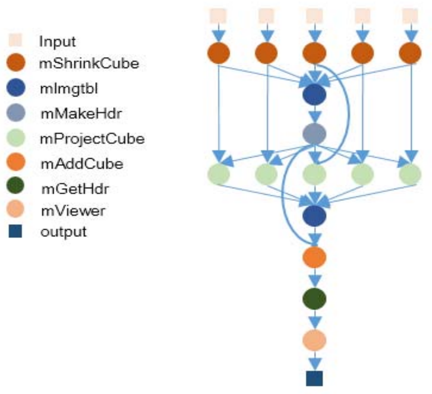


Fig. 6. Montage GALFA workflow

In our experiments, we acquired a set of five data cubes, released as part of the Galactic Arecibo L-band Feed Array HI (GALFA-HI [11]) survey, into a mosaic in three major stages as shown in Figure 6[12]. In the first step, an output data is generated by averaging and reducing according to a predetermined planes value. Next, the shrunk images were re-reflected to map the input pixel space to sky coordinates and to the output pixel space. Finally, the re-reflected images are acquired to produce the final mosaic. Overall, the GALFA Montage workflow reads 27.5GB of input data and at least 3.0GB of intermediate data and writes at least 3.5GB of output data. The size of the data depends on the planes averaged during the reduction process. In this experiment, experiments were performed using 15 planes at the mShrinkCube stage.

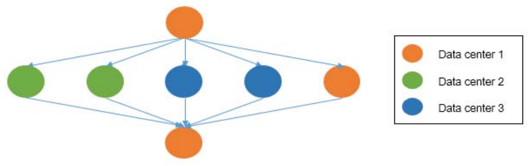
B. Experiment Environments

This experiment uses OpenStack [13], a cloud management platform (CMP) that defines and manages a large pool of computing, storage, and networking resources used in large HPC science experiments. Our experiment environment consists of one controller node and three compute nodes on the same local network. And we simulated nodes as data centers. The nodes in our system have a total RAM of 16GB and 8CPU cores for the controller node and total RAM of 32GB and 12CPU cores for each of the three compute nodes respectively. Each of these server systems is operated by Ubuntu 14.04.5 Trusty Tahr.

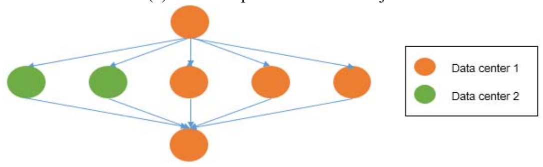
C. Experiment Strategy

If the state of available resources changes during task execution, the execution time of Montage application is measured using OpenStack to compare the data transfer time and the execution time of the data before replacement (experiment 1) and after replacement (experiment 2) considering the state of resources.

Experiment 1 generates two VMs for each of the three data centers, and assign tasks to the generated six VMs. Experiment 2 assumes that resources previously allocated during task execution were able to allocate new tasks at the end of the task, and places tasks as Experiment 1 in initial stage. In mProjectCube module, create three VMs of each of the two data centers, and replace tasks to the generated six VMs. The mProjectCube module has the most intermediate data among the various modules of Montage and takes the longest execution time, thus reducing the number of data movement between the data centers. Figure 7 shows the modules from mMakeHdr to mImgtbl, and shows states before and after replacing tasks to data centers in the mProjectCube module.



(a) Before replacement of mProjectCube



(b) After replacement of mProjectCube

Fig. 7. States of before and after replacing

D. Experiment Results

In Fig. 7, arrows indicate the movement between data centers, as a result, the number of movements between data centers before and after replacement in the mProjectCube module was 8 in Fig. 7 (a) and decreased to 4 in Fig. 7 (b).

In the Experiment 2, the data transfer time between the mMakeHdr module and the mProjectCube module decreased by 7.9% on average compared to Experiment 1. And in the Experiment 2, data transfer time between the mProjectCube module and the mImgtbl module was reduced by 50% on the average compared to the Experiment 1.

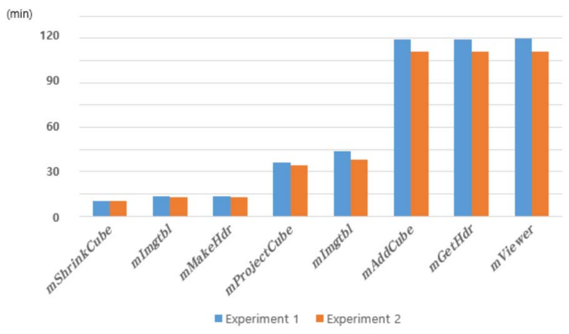


Fig. 8. End time of each module

Figure 8 shows the end-time of each module from the first execution start time, and is shown as an average graph. In the modules of mSrinkCube, mImgtbl, and mMakeHdr before replacement, the end-time of each module was similar in Experiment 1 and Experiment 2. And in the replaced mProjectCube module, the number of movements of data between data centers decreased as shown in Fig. 6. As a result, in the mProjectCube module, Experiment 2 decreased by an average of 5.7% as compared with Experiment 1. In the modules of mAddCube, mGetHdr, and mViewer, Experiment 2 decreased by 7.04%, 7.05%, and 7% on average compared to Experiment 1. And the total execution time taken from the start to the end of the Experiment 2 decreased by an average of 7% compared with Experiment 1.

In this paper, we show that the application execution time and data transfer time are reduced when the data is replaced considering the dynamically changing resource state and data-intensive task during execution. In Addition, the number of data centers used decreased and the execution time of the application decreased at the same time. Therefore, we can see that replacement at the point of data-intensive tasks of execution is very important factor in this paper.

V. Conclusions and Future Work

In this paper, we proposed an adaptive data placement strategy for a data-intensive scientific application based on the paper [4], which considered data dependency. For better resource utilization, we added a feature to the strategy of watching the state of resource dynamic change during the execution of tasks, dependency among datasets and tasks. That leads to be recalculation of dependency among datasets and tasks; and replacement of location of data just before executing data-intensive tasks, which are part of an application.

Experiment results show that the proposed strategy is efficient because data transfer time and execution time of an application is reduced as a result of applying our data replacement strategy

We will extend this paper by applying this strategy to various data-intensive applications over diverse cloud resource environment in the future.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2015M3C4A7065646)

References

[1] [1] E. Deelman, A. Chervenak, Data management challenges of data-intensive scientific workflows, in: IEEE International Symposium on Cluster Computing and the Grid, pp. 687-692.(2008)

[2] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny, Pegasus: Mapping scientific workflows onto the grid, in: European Across Grids Conference, pp. 11-20.(2004)

[3] CFD, www.cfd-online.com, viewed 2009

[4] D.Yaun, Y.Yang, X.Liu and J.Chen, A data placement strategy in scientific cloud workflows, J.Future Generation Computer Systems, 26, 8(2010)

[5] Q.Zhao, Xiong, C., Zhang, K., Yue, Y., & Yang, J. A Data Placement Algorithm for Data Intensive Applications in Cloud. International Journal of Grid and Distributed Computing, 9(2), 145-156 (2016)

[6] S. Agarwal et al., Volley: Automated Data Placement for Geo-Distributed Cloud Services, Proc. 7th USENIX Conf. Networked Sys. Design and Implementation (2010)

[7] Q. Xu, Z. Xu, and T. Wang, A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing, International Journal of Intelligence Science 5 (2015)

[8] A. Weiss, Computing in the Cloud, ACM Networker, **(2007)**, vol. 11, pp. 18-25.

[9] W.T. McCormick, P.J. Sehweitzer, T.W. White, Problem decomposition and data reorganization by a clustering technique, Operations Research, 20, pp. 993-1009 (1972)

[10] Mosaic image generation engine in astronomy field (Montage GALFA), http://montage.ipac.caltech.edu/docs/cubemosaicstutorial.html

[11] Peek, J.E.G., et al.: The GALFA-HI survey: data release 1. Astrophys. J. Suppl. 194(2), 20 (2011)

[12] Choi, Jieun, Theodora Adufu, and Yoonhee Kim. "Data-locality aware scientific workflow scheduling methods in HPC cloud environments." International Journal of Parallel Programming (2017): 1-14.

[13] OpenStack, http://www.OpenStack.org