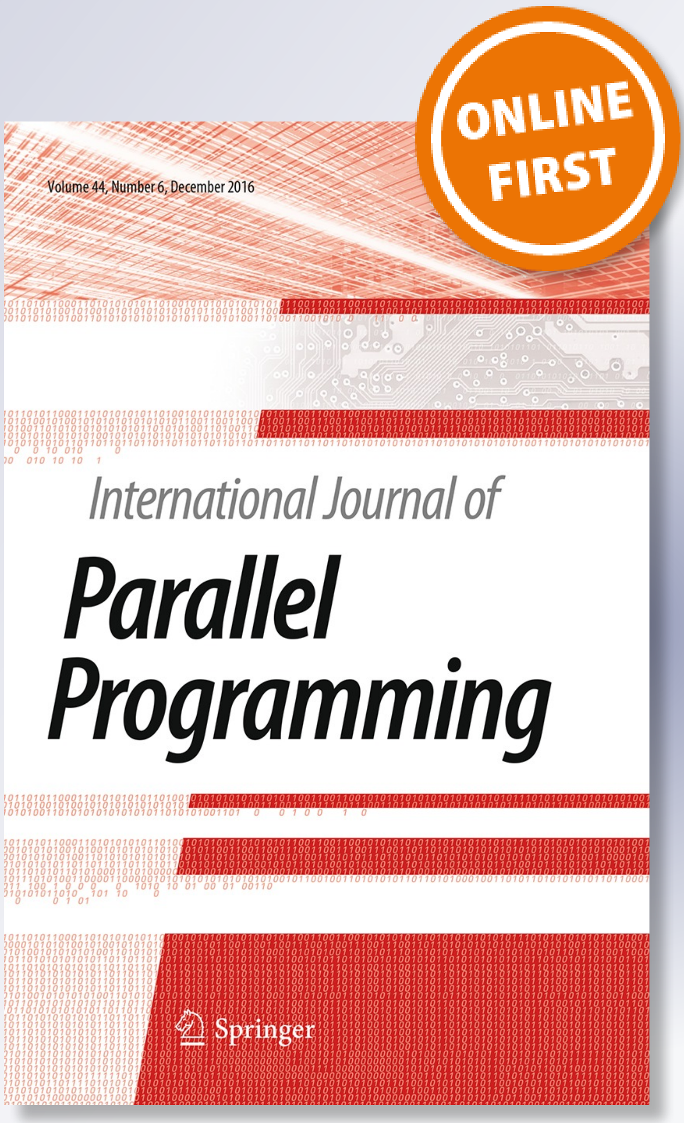
Data-Locality Aware Scientific Workflow Scheduling Methods in HPC Cloud Environments

Jieun Choi, Theodora Adufu & Yoonhee Kim

International Journal of Parallel Programming

ISSN 0885-7458

Int J Parallel Prog
DOI 10.1007/s10766-016-0463-0



Int J Parallel Prog
DOI 10.1007/s10766-016-0463-0

Data-Locality Aware Scientific Workflow Scheduling Methods in HPC Cloud Environments

Jieun Choi[1] · Theodora Adufu[1] · Yoonhee Kim[1]

Received: 1 April 2016 / Accepted: 22 September 2016

Abstract Efficient data-aware methods in job scheduling, distributed storage management and data management platforms are necessary for successful execution of data-intensive applications. However, research about methods for data-intensive scientific applications are insufficient in large-scale distributed cloud and cluster computing environments and data-aware methods are becoming more complex. In this paper, we propose a Data-Locality Aware Workflow Scheduling (D-LAWS) technique and a locality-aware resource management method for data-intensive scientific workflows in HPC cloud environments. D-LAWS applies data-locality and data transfer time based on network bandwidth to scientific workflow task scheduling and balances resource utilization and parallelism of tasks at the node-level. Our method consolidates VMs and consider task parallelism by data flow during the planning of task executions of a data-intensive scientific workflow. We additionally consider more complex workflow models and data locality pertaining to the placement and transfer of data prior to task executions. We implement and validate the methods based on fairness in cloud environments. Experimental results show that, the proposed methods can improve performance and data-locality of data-intensive workflows in cloud environments.

Keywords Data-aware scheduling · Data-intensive application · Data-locality · Cloud · Scientific workflow

✉ Yoonhee Kim
yulan@sookmyung.ac.kr

Jieun Choi
jechoi1205@sookmyung.ac.kr

Theodora Adufu
theodora@sookmyung.ac.kr

[1] Department of Computer Science, Sookmyung Women's University, Seoul 140-742, Korea

1 Introduction

Available computing infrastructures for an application are becoming increasingly diverse such as grid, cluster, container clusters and cloud. Amongst these infrastructures, cloud computing has rapidly become a widely adopted paradigm for scientific experiments, big-data analytics, and data visualization applications. In such highly distributed computing environments, traditional scientific applications, which are predominantly compute-intensive, require high-performance computing resources to facilitate optimum execution of tasks. Consequently, existing scheduling techniques have given priority to optimizing resource provisioning in order to maximize efficiency for such executions. Some scheduling techniques [1,2] use Path Clustering Heuristic (PCH) algorithms to find the critical paths for task executions on resources and schedule tasks accordingly. The other methods [3] reconfigure VM resources with an attempt to satisfy user-specified Service Level Agreements (SLAs) as well as enhance overall system performance and utilization at minimum overhead. However, increasing data volumes generated during task executions of recent scientific applications reveal that these scheduling techniques do not adequately apply data-aware methods for optimal performance.

Executing data-intensive workflows predominantly involve I/O-intensive tasks since data is continuously read from and written to assigned storage locations. Increasing volumes of input and output data, as well as large data-sets per task generated at the intermediary stages of executions heightens the need to incorporate more data-aware methods in scheduling techniques. As a result, scheduling data-intensive scientific workflows has become a key research concern as more HPC scientific applications are deployed on HPC clouds and on exascale computing platforms. Existing methods [1,2,4] do not consider data locality for optimal execution of data-intensive scientific workflows. However, data locality can drastically decrease network usage and application execution time and improve performance of data-intensive scientific workflows. Hence, efficient data-aware methods in job scheduling, distributed storage management and data management platforms are imperative for successful execution of data-intensive scientific workflows.

We propose a novel Data-Locality Aware Workflow Scheduling (D-LAWS) technique, which applies data locality, data transfer time based on network bandwidth, VM consolidation, fairness and user-specified SLA-sensitivity to data-intensive scientific workflow task scheduling at the node-level. D-LAWS maximizes resource utilization and parallelism of tasks and compares data transfer times of VMs based on network bandwidth between VMs. Our methods consolidate VMs and consider task parallelism by data flow during the planning of task executions of a data-intensive scientific workflow. We additionally consider more complex workflow models and data locality regarding the placement and transfer of data prior to task executions. We implement and validate our methods based on fairness in a cloud environment using OpenStack [5] which is a cloud operating system that controls compute, storage and networking resources in a data-center. Experimental results show that, the proposed method can improve performance and data-locality of data-intensive scientific workflows in cloud environments.

Novel contributions of this paper are as follows:

- We propose a data-locality aware workflow scheduling technique, called D-LAWS, which includes a data-aware workflow scheduling algorithm and a resource consolidation method considering data locality.
- D-LAWS considers not only a critical path of a workflow but also data locality and data transfer time based on network bandwidth which is crucial in a distributed environment with heterogeneous resources.
- The proposed method also considers task parallelism by data flow during the planning of tasks executions in order to maximize resource utilization using a VM resource consolidation algorithm.
- We evaluated our method in a private cloud computing environment and compare results to two other workflow scheduling techniques.

The rest of this paper is organized as follows: Sect. 2 presents the related work and Sect. 3 describes our Data-Locality Aware Workflow Scheduling (D-LAWS) techniques. In Sect. 4, we explain our experiment environments. We finally conclude this paper in Sect. 5.

2 Related Work

There has been a wide variety of research in the area of data-aware techniques for data-intensive applications. In this section, we present and analyze works related to data-aware methods for data-intensive applications, including data-aware task scheduling and locality-aware resource placement methods.

In [6], the researchers present a task scheduling strategy to mitigate interference whilst preserving task-data locality for MapReduce applications. However, the research ignores network factors with the assumption that data exchange between co-hosted VMs is as efficient as local data accesses. Also, Phan et al. [7], explore an Earliest Deadline First (EDF) scheduling technique for real time cloud-based data processing. They schedule tasks considering earlier deadlines first, as well as a weighting factor of the scheduler which indicates how important data locality is. The weighting factor indicating data locality is however chosen manually; without consideration to factors such as network or application characteristics. Zaharia et al. [8] develop the Longest Approximate Time to End (LATE) scheduling algorithm which uses estimated finish times to speculatively execute tasks in order to improve their response times. This algorithm however assumes that tasks make progress at a roughly constant rate which is untrue for most scientific workflows which have a variation of both long and short tasks. In another paper [9], they proposed a delay scheduling algorithm to postpone a scheduled job for a few seconds if it cannot launch a local task. However, scheduling is based on the assumption that, tasks are short lived thereby rendering the algorithm ineffective for long running jobs. Wang et al. [10] propose a data-aware work stealing technique to optimize both load balancing and data-locality. They consider fully-distributed architecture, so all schedulers are fully-connected and receive workloads to schedule tasks to local executors. They determine the ultimate location of data stored using a hashing method in order to ensure the best write-locality. However, this structure results in a significant communication overhead for a plurality of schedulers. When a scheduler schedules a task, it considers only data size and task length but does not consider network bandwidth and heterogeneous resources capacity. However, data transfer time is dependent on network bandwidth at the node-level or system-level in computing environments.

We also present related research on scientific workflow scheduling methods [1–4]. Bittencourt et al. [1] use Path Clustering Heuristic(PCH) algorithms to find the critical paths for task executions on resources and schedule tasks accordingly. They consider public, private and hybrid clouds. In [3], they employ a Load Vector per task to estimate the relative running time of the VM instance on which the task is run. An instance acquisition lag is defined by which VMs are either acquired or released during workflow execution. This improves instance hour consolidation and improves costs however, data transfer costs between VM's on different nodes is not considered. Also, inaccurate estimations of VM acquisition lag times result in significant performance overheads. In [2,4], there are proposed workflow scheduling algorithms and auto-scaling methods for hybrid computing environments. However, these papers do not consider data-locality at all. There is also research on resource placement and storage-aware scheduling techniques considering the location of the storage. Thaha et al. [11] propose a location aware cluster provisioning strategy for cloud based virtual Hadoop clusters to improve the movement of data by placing the clusters' VMs in compute hosts with shortest distance to the storage node. Their technique not takes into account the data generated dynamically because they only consider and assume the input data prepared by the user in advance. Bryk et al. [12] propose a novel dynamic scheduling algorithm that is aware of the underlying storage system. They develop a global storage model which features the ability to dynamically calculate bandwidth and supports a configurable number of replicas. Therefore, it is possible to calculate a relatively exact file transfer time through dynamic bandwidth calculation. They place and do work with the data in the cache before all VM executions. When scheduling a task, a scheduler confirms whether or not the data is in the VM's local cache for a locality purposes. They perform simulations to verify their algorithm considering various storage systems such as no storage system, in-memory storage, distributed storage, NFS storage. However, it's not easy to apply their techniques when the data-size gets larger.

We propose a novel technique, D-LAWS (Data-Locality Aware Workflow Scheduling), which considers data locality and applies consolidation of VMs for scientific workflow task scheduling. D-LAWS balances data-locality, system utilization, and parallelism by tasks and compares data transfer time based on network bandwidth between VMs. In hybrid environments, our technique can facilitate an efficient selection of resources based on application characteristics as well as comparisons of data transfer costs to each available resource.

3 Data-Locality Aware Workflow Scheduling (D-LAWS) Technique

In this section, we present our algorithms. The Data-Locality Aware Workflow Scheduling (D-LAWS) techniques include a data-aware workflow scheduling algorithm and a resource consolidation method. Two algorithms are shown in Algorithms 1 and 2. The key notations used in the algorithms are listed in Table 1.

Table 1 Notations

Notation	Description
$\mathbb{W}$	A workflow application
t_i	An ith task in a workflow $\mathbb{W}$
t_i^{parent}	A set of precedence tasks of task t_i
$\lvert t_i^{parent} \rvert$	The number of precedence tasks of t_i
T_i	A set of tasks which have same parents and children tasks
$vmSet$	A set of VMs which have input data of t_i
$idleVMSet$	A set of VMs which during certain period
CP	A critical path of $\mathbb{W}$
VM_{jh}	jth VM in host h
$\forall$VM	All available VMs on which t_i can run at that time
D_{ik}	kth input data of t_i
DTT	Data transfer time
Min_C	Required minimum core capacity of VM
Min_R	Required minimum ram capacity of VM
Min_D	Required minimum disk capacity of VM
EFT_t_i	Estimated finish time of t_i
EST_t_i	Estimated start time of t_i
ET_t_i	Execution time of t_i
EST_VM_{jh}	Estimated start time of VM_{jh}
$LFT\_t_i^{parent}$	Estimated finish time of a precedence task which has latest finish time

3.1 Data-Aware Workflow Scheduling Algorithm

Algorithm 1 shows a scheduling procedure for data-intensive scientific workflow in a cloud computing environment. Algorithm 1 describes a data locality aware task scheduling considering data size, network bandwidth, and resource utilization. It starts when a workflow comes to system with SLA (Service Level Agreements) which includes a desired deadline for the workflow and minimum capacity of a VM such as core, memory, and disk capacity by a user.

First, estimated finish time (EST_t_i), execution time (ET_t_i) and estimated start time(EFT_t_i) of all tasks in a workflow are initialized to zero (line 1) and then the scheduler schedules each task t_i into VMs. Next, all tasks on the critical path are scheduled on the same resource, which can execute all tasks in a critical path (lines 3–8). If the task t_i is the first task in the critical path, t_i is scheduled to VM_{jh} which is part of the set of all available VMs ($\forall$VM) on which all tasks in a critical path can be executed within a given deadline(D). If there is no available VM, the scheduler creates a VM on which all tasks in the critical path can run within a deadline and satisfy a required minimum capacity of resources (line 5). After that, other tasks in the critical path are scheduled to the currently selected resource (line 7). By placing all tasks in a critical path on the same resource, there would be no transfer of data from one VM to another for tasks in the critical path thus minimizing the cost of data transfer to zero for those tasks in the critical path. Since there is no data transfer, DTT for tasks in the critical path is zero (0) and thus omitted in our proposed algorithm. The scheduler then schedules tasks that are not in the critical path (lines 9–17) with consideration to EFT of previously related tasks. In lines 9–10 of the algorithm, if the set size of a precedence task ($|t_i^{parent}|$) of a task t_i, equals zero then the task is an entry node and the scheduler finds a VM in the same way as line 5. However, in case that task t_i is not in the critical path and has parent tasks, the task's EST is set to the sum of the longest finish time of parent tasks and the data transfer time (DTT). After that, the scheduler finds VMs which already have input data of t_i (lines 13–15). Finally, the task is scheduled to the most suitable VM_{jh} and EFT is calculated (line 18). In this algorithm, data transfer time (DTT) for a task can be defined as shown in equation 1.

Algorithm 1 Data-aware Workflow Scheduling Algorithm

```
Input: a Workflow $\mathbb{W}$, SLA(deadline D, $Min_C$, $Min_R$, $Min_D$)
1: Set $EFT\_t_i$, $ET\_t_i$, $EST\_t_i$ = 0;
2: for each $t_i$ in $\mathbb{W}$ do
3:   if $t_i \in$ CP then
4:     if $|t_i^{parent}| == 0$ then
5:       $VM_{jh} \leftarrow$ FindVM($EST\_t_i$, SLA, $\forall$VM) on which all tasks $\in$ CP within D;
6:     else
7:       $VM_{jh} \leftarrow$ the VM on which first task $\in$ CP is scheduled;
8:     end if
9:   else
10:    if $|t_i^{parent}| == 0$ then
11:      $VM_{jh} \leftarrow$ FindVM($EST\_t_i$, SLA, $\forall$VM);
12:    else
13:      $EST\_t_i \leftarrow LFT\_t_i^{parent}$ + DTT;
14:      vmSet $\leftarrow$ find VMs with input data of $t_i$;
15:      $VM_{jh} \leftarrow$ FindVM($EST\_t_i$, SLA, vmSet);
16:    end if
17:  end if
18:  $EFT\_t_i \leftarrow EST\_t_i + ET\_t_i$;
19: end for
20: Perform resource consolidation;
Output: Scheduling decision $\mathbb{SD}$ = {($t_i$, $VM_{jh}$) | $i, j, h$ = 0,1,..., N }
```

$$DTT\_t_i = \sum_{t=0}^{|vmSet|} \frac{DataSize(t)}{NetworkBandwidth} \tag{1}$$

3.2 VM Consolidation Algorithm

Data-aware scheduling methods for data-intensive scientific workflows aim at minimizing the time taken by tasks to read inputs during workflow executions. The VM consolidation algorithm in [3] applies a parallelism reduction method of deadline assignment algorithm to a simple workflow. We consider task parallelism by data flow during the planning of task executions of a data-intensive scientific workflow. We additionally consider more complex workflow models and data locality regarding the placement and transfer of data prior to task executions.

In the resource consolidation algorithm, T_i is a set of tasks which have same parents and children. In lines 2–6, if all tasks of T_i can be executed on a single VM_{jh}, when they are combined then re-assign all tasks to that VM, VM_{jh}. This provides a parallelism reduction method and efficient VM utilization. Algorithm 2 describes rescheduling a task into idle VM (lines 7–14) to improve resource utilization. When the scheduler determines which tasks to be rescheduled, it calculates estimated execution time, ET_t_i from the earliest start time, EST_t_i and compares the resulting time and the data transfer time (DTT), to the EST_VM_{jh} of idle VMs. If the cost of starting a new VM is greater than the cost of transferring tasks to another VM, then find a suitable VM and reassign tasks.

Algorithm 2 Resource consolidation algorithm

```
Input: Initial scheduling decision $\mathbb{SD} = \{(t_i, VM_{jh}) \mid i, j, h = 0,1,\ldots, N\}$
1: Set $T_i = \{ t_i$ have same parents and children $\}$
2: for each $T_i$ do
3:   if $\forall t_i \in T_i$ can be executed on a single $VM_{jh}$ then
4:     $VM_{jh} \leftarrow$ Re-assign $\forall\, t_i$;
5:   end if
6: end for
7: for each $t_i$ do
8:   if there are idle VMs during $EST\_t_i + ET\_t_i$ then
9:     idleVMSet ← idle VMs
10:    if $(EST\_t_i + ET\_t_i + DTT \leftarrow EST\_VM_{jh})$ then
11:      $VM_{jh} \leftarrow$ FindVM( $EST\_t_i$, idleVMSet)
12:    end if
13:  end if
14: end for
Output: Updated scheduling decision $\mathbb{SD} = \{(t_i, VM_{jh}) \mid i, j, h = 0,1,\ldots, N\}$
```

4 Experiments

Experiments that validated our D-LAWS technique are presented in this section. First, we describe the system architecture and target applications, and subsequently present the experimental setting along with experimental results.

4.1 Experiment Environments

We use OpenStack [5], a cloud management platform (CMP) that provisions and manages large pools of compute, storage and networking resources used for large-scale HPC scientific experiments. OpenStack, has multiple services which improve the effective management of cloud resources. This include the Nova Compute and Cinder which is used for the creation of new instances or storage volumes. The

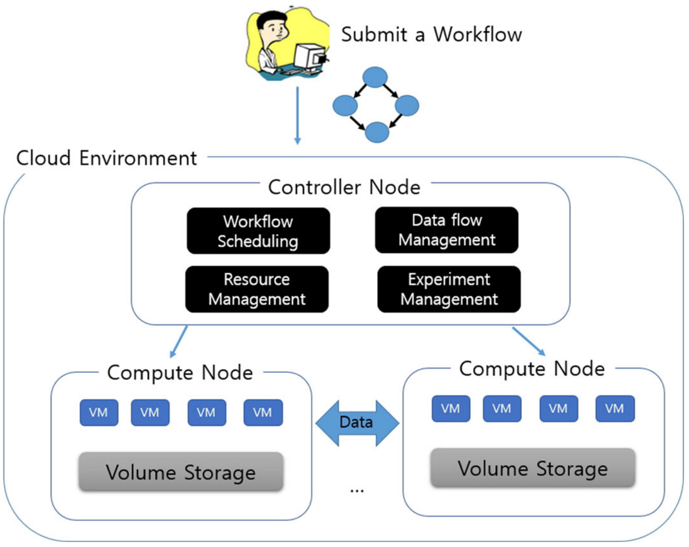


Fig. 1 System architecture

resources of each VM such as RAM, disk and vCPUs, created using Nova Compute is defined by using default or user-defined install templates or flavors [13]. Memory and CPU can be over-committed on compute nodes enabling access to more computing resources though this affects the performance of the instances [14]. This notwithstanding, OpenStack is widely deployed for the provisioning and management of virtual cloud resources for HPC applications. As shown in Fig. 1, our experiment environment consists of one controller node and three compute nodes on the same local network. The nodes in our system have a total RAM of 8 GB and 8 CPU cores for the controller node and total RAM of 32 GB and 12 CPU cores for each of the three compute nodes respectively. Each of these server machines are operated by Ubuntu 14.04 Trusty Tahr.

4.2 Data-Intensive Scientific Workflow

Scientific workflows can be represented as Direct Acyclic Graphs(DAG), in which the vertex are small discrete tasks and the edges represent the data flows from on task to another. In this paper, we implement our algorithms using a data-intensive workflow, Montage [15]. Montage is an Astronomical Image Mosaic Engine for creating composite FITS (Flexible Image Transport System) mosaics using multiple astronomical images.

In our experiments, we aggregated a set of five data cubes, released as part of the Galactic Arecibo L-band Feed Array HI (GALFA-HI [16]) survey, into a mosaic in three major stages as shown in Fig. 2. First, the input data cubes were shrunk by averaging every 5, 10 and 15 planes respectively in the wavelength axis. In this manner, shrinking by averaging every 10 planes for instance, produces larger intermediary and output data files all throughout the execution of the workflow than shrinking by every

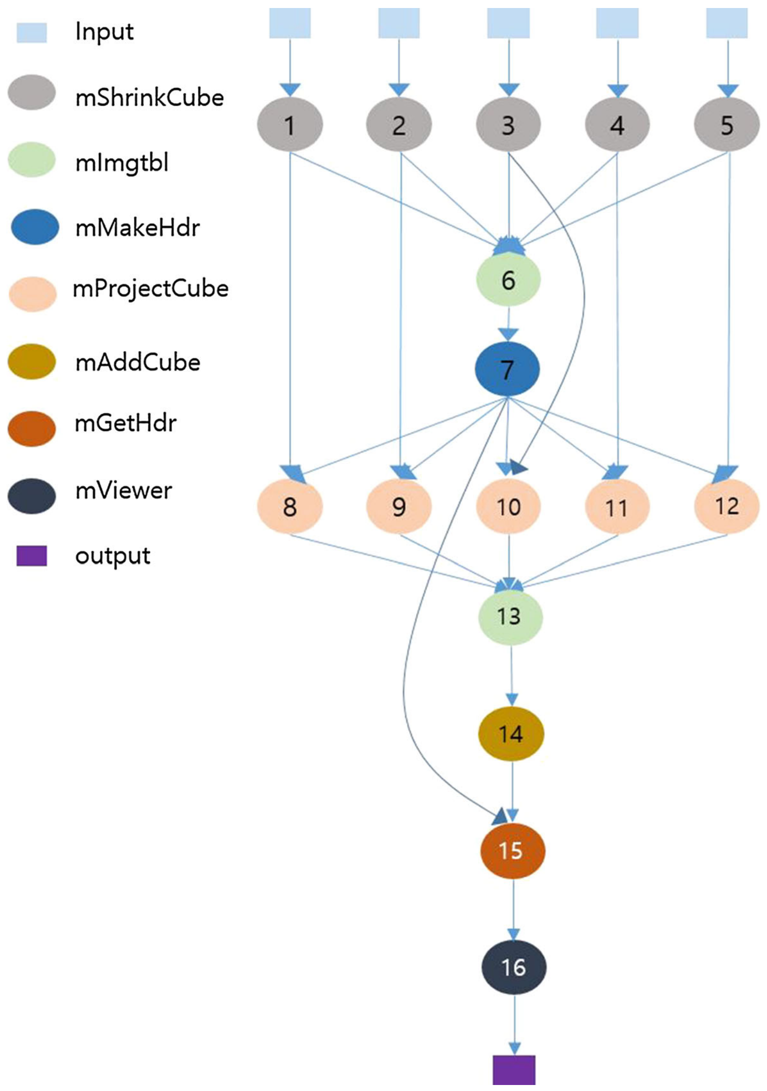


Fig. 2 Data-intensive scientific workflow with data flow

15 planes. Next, the shrunk images were re-projected to map the input pixel space to sky coordinates and to the output pixel space. Finally, the re-projected images are aggregated to produce the final mosaic.

In all, the GALFA Montage workflow in our experiments reads 27.5 GB of input data and a minimum of 3.0 GB of intermediary data and writes a minimum of 3.5 GB of output data. The data size is dependent on the number of planes which are averaged during the shrink process. Consequently, shrinking by averaging greater number of planes produces smaller intermediary and output data files. The output of one process

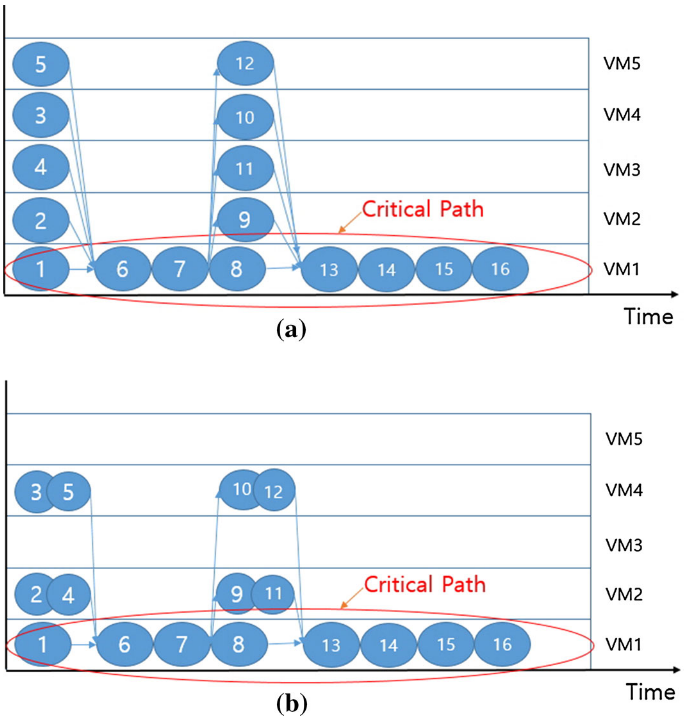


Fig. 3 Result of VM consolidation for T_i. **a** Before VM consolidation, **b** after VM consolidation

becomes the input to the next implying that significant amount of execution time is spent on input/output (I/O) operations.

Figure 3 describes a scheduling result of our data-aware workflow scheduling algorithm, as shown in Sect. 3.1. There are five VMs in three different nodes. We used two instance types of VM, medium and large, with 4 GB RAM each and having 2core and 4core correspondingly. Among the VMs, three VMs (VM1, VM3, VM5) are medium type and two VMs (VM2, VM4) are large instance type. VM1 is in node 1, VM2 and VM3 are in node 2 and, VM4 and VM5 are in node 3. The arrows represent data flows.

For the case of Fig. 3a, there is one critical path with tasks (1, 6, 7, 8, 13, 14, 15, 16) scheduled to VM1. As discussed in the notations (Table 1), T_i represents a set of tasks which have same parent and children tasks. In Fig. 3a, T_1 has five tasks (1–5) and also T_2 has five tasks (8–12). According to Algorithm 2, the scheduler can perform VM consolidation for T_i without affecting other tasks. As a result, we see the scheduling of tasks from consolidating VMs as shown in Fig. 3b.

4.3 Experiment Results

We evaluate the performance of the D-LAWS technique on a private cloud environment as shown in Sect. 4.1. We used a data-intensive scientific workflow as shown in Sect. 4.2. And we consider three shrink sizes: averaging every 5, 10, 15 planes

Table 2 Data sizes for 5,10,15 planes

Data description	15 Planes	10 Planes	5 Planes
Final mosaic FITS	3.5 G	5.1 G	11 G
Final mosaic area	26 M	26 M	26 M
Reprojected data cube	2.5 G	3.7 G	7.4 G
Reprojected data area	18.8 M	18.8 M	18.8 M
Image tables	5.8 K	5.8 K	5.8 K
Header files	393 B	393 B	393 B
Shrunk data cube	341 M	511 M	1023 M
Input data cube	27.5 G	27.5 G	27.5 G

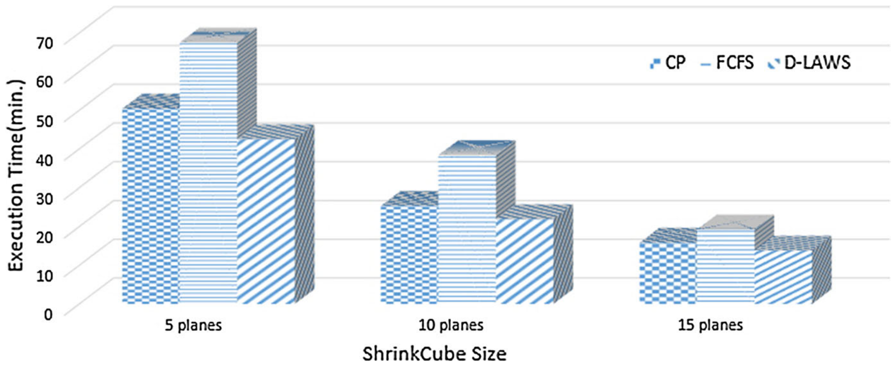


Fig. 4 Execution time by CP, FCFS, D-LAWS for 5, 10, 15 planes

respectively. Table 2 shows the description of total input, intermediary and output data produced during the execution of Montage GALFA workflow and their corresponding data sizes for 5, 10, 15 planes respectively. We compare the proposed technique which is represented as D-LAWS in the graph with CP, which schedules a workflow considering only critical path [1], and FCFS which schedules a workflow regarding task priority and queue order. The experiment results are shown in Figs. 4, 5, 6 and 7.

Figure 4 shows the execution times of CP, FCFS and D-LAWS for data cubes shrunk over 5, 10 and 15 planes. Corresponding to the data sizes of shrunk images used during the executions as shown in Table 2, 5 planes take the longest execution time. The D-LAWS technique speed-up values of overall execution times of a workflow. For instance, comparing D-LAWS relative to CP, the speed-up values are 18.06, 15.81 and 14.52 % respectively for 5, 10 and 15 planes with an average speed-up of 18 %. Meanwhile, D-LAWS has an average speed-up value of 57.63 % relative to FCFS with specific values of 58.83, 75.06 and 39.01 % respectively for each of the 3 planes. The improvement in speed-up times of D-LAWS is a result of considerations for data locality and data transfer times.

In Fig. 5, we see resource utilization per given time of D-LAWS relative to CP and FCFS. Initially, the resource utilization decreases because there are many tasks as well as parallelism. Between 3 and 4 points in time, the overall trend of resource utilization

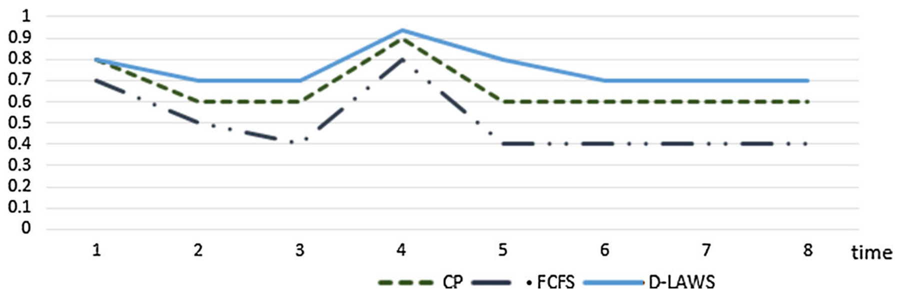


Fig. 5 Resource utilization

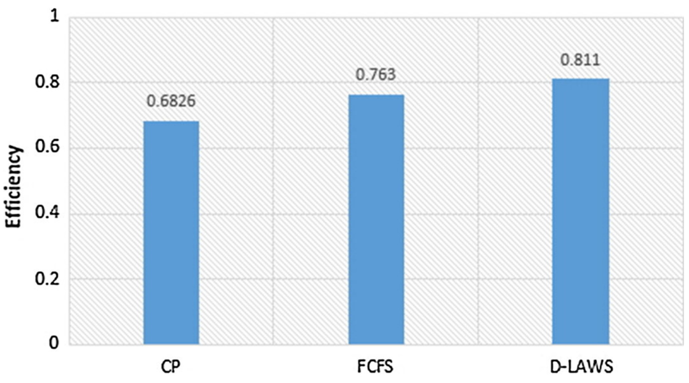


Fig. 6 Efficiency

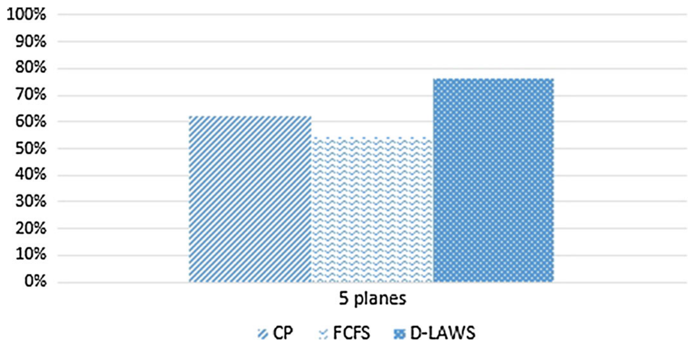


Fig. 7 Data locality

increases steadily and then decreases after time point 4. This is due to the increase amount of computations performed during the execution of mProjectCube module in our data-intensive workflow. And when D-LAWS is compared with FCFS, it shows up to 30 % more efficient resource utilization. When compared to CP, D-LAWS shows 20 % higher efficiency in resource utilization. The overall resource utilization of D-LAWS is the best among the three methods with the reason being that, D-LAWS could consolidate VMs according to Algorithm 2 as shown in Section 3.2.

Figure 6 describes efficiency of D-LAWS relative to CP and FCFS. Efficiency refers to the ratio of time that the system is executing tasks [10] with 1 as an ideal value from the system view. From Fig. 5, we see efficiency values as 0.68, 0.76, 0.81 corresponding to CP, FCFS, and D-LAWS respectively. When using the D-LAWS technique the scheduler could reduce the data transfer time considering data-locality based on network bandwidth and data size while the system executes tasks. Therefore, the experiments perform most efficiently with the D-LAWS technique.

Figure 7 shows data locality of three methods. We see that for shrink cube size averaged over 5 planes, D-LAWS improved by approximately 37 % with respect to other techniques. The reason is that D-LAWS considered data locality regarding the placement and transfer of data prior to task executions. Even though other methods considered important factors such as resource capacity and fastest estimated start time of VMs, which can increase throughput and speed, without considering data-locality their performance is sub-optimal. In contrast, our technique D-LAWS improves the overall execution time and resource utilization while minimizing scheduling overhead for data-intensive scientific workflow.

5 Conclusions

Efficient data-aware methods in job scheduling, distributed storage management and data management platforms are necessary for successful execution of data-intensive applications. In this paper, we propose a Data-Locality Aware Workflow Scheduling (D-LAWS) technique and a locality-aware resource management method for data-intensive scientific workflows on HPC cloud environments. D-LAWS applies data-locality and data transfer time based on network bandwidth to scientific workflow task scheduling and balances resource utilization and parallelism of tasks. Our methods consolidate VMs and consider task parallelism by data flow during the planning of task executions of a data-intensive scientific workflow. We additionally consider more complex workflow models and data locality regarding the placement and transfer of data prior to task executions. We implement and validate the methods based on fairness in cloud environments. Experimental results show that, the proposed method can improve performance and data-locality of data-intensive workflows in cloud environments. In the future, we will consider hybrid cloud environments and also we will experiment for scientific applications with variant characteristics.

Acknowledgments 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. Bittencourt, L.F., Madeira, E.R.M.: HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. J. Internet Serv. Appl. **2**, 207–227 (2011)
2. Ahn, Y., Kim, Y.: Auto-scaling of virtual resources for scientific workflows on hybrid clouds. In: ScienceCloud '14 Proceedings of the 5th ACM Workshop on Scientific Cloud Computing (pp. 47–52). (2014)

3. Mao, M., Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (p. 49). ACM (2011)
4. Choi, J., Ahn, Y., Kim, S., Kim, Y., Choi, J.: VM auto-scaling methods for high throughput computing on hybrid infrastructure. J. Clust. Comput. **18**, 1063–1073 (2015)
5. OpenStack, http://www.OpenStack.org
6. Bu, X., Rao, J., Xu, C.-Z.: Interference and locality-aware task scheduling for MapReduce applications in virtual clusters. In: HPDC '13 Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing (pp. 227–238). (2013)
7. Phan, L.T., Zhang, Z., Zheng, Q., Loo, B.T., Lee, I.: An empirical analysis of scheduling techniques for real-time cloud-based data processing. In: SOCA '11 Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications (2011)
8. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving MapReduce performance in heterogeneous environments. In: OSDI'08 Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (pp. 29–42). (2008)
9. Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: EuroSys '10 Proceedings of the 5th European Conference on Computer Systems (pp. 265–278). (2010)
10. Wang, K., Qiao, K., Sadooghi, I., Zhou, X., Li, T., Lang, M., Raicu, I.: Load-balanced and locality-aware scheduling for data-intensive workloads at extreme scales. J. Concurr. Comput. Pract. Exp. (CCPE) **28**, 70–94 (2015)
11. Thaha, A.F., Singh, M., Amin, A.H.M., Ahmad, N.M., Kannan, S.: Hadoop in OpenStack: Data-location-aware cluster provisioning. In: Information and Communication Technologies (WICT), 2014 Fourth World Congress on (pp. 296–301). (2014)
12. Bryk, P., Malawski, M., Juve, G., Deelman, E.: Storage-aware algorithms for scheduling of workflow ensembles in clouds. J. Grid Comput. **14**, 359–378 (2015)
13. Flavors, http://docs.openstack.org/openstack-ops/content/flavors.html
14. Overcommiting on compute nodes, http://docs.openstack.org/openstack-ops/content/compute_nodes.html
15. Montage, http://montage.ipac.caltech.edu/
16. Peek, J.E.G., et al.: The GALFA-HI survey: data release 1. Astrophys. J. Suppl. **194**(2), 20 (2011)