Measuring Data Locality Ratio in Virtual MapReduce Cluster Using WorkflowSim

Peerasak Wangsom
School of Information Technology
King Mongkut's University of Thonburi
Bangkok, Thailand
peerasak.w@mail.kmutt.ac.th

Kittichai Lavangnananda
School of Information Technology
King Mongkut's University of Thonburi
Bangkok, Thailand
kitt@sit.kmutt.ac.th

Pascal Bouvry
Faculty of Sciences, Technology and Communication
University of Luxembourg
Luxembourg
pascal.bouvry@uni.lu

***Abstract*— The data locality is significant factor which has a direct impact on the performance of MapReduce framework. Several previous works have proposed alternative scheduling algorithms for improving the performance by increasing data locality. Nevertheless, their studies had focused the data locality on physical MapReduce cluster. As more and more deployment of MapReduce cluster have been on virtual environment, a more suitable evaluation of MapReduce cluster may be necessary. This study adopts a simulation based approach. Five scheduling algorithms were used for the simulation. WorkflowSim is extended by inclusion of three implemented modules to assess the new performance measure called 'data locality ratio'. Comparison of their results reveals interesting findings. The proposed implementation can be used to assess 'data locality ratio' and allows users prior to efficiently select and tune scheduler and system configurations suitable for an environment prior to its actual physical MapReduce deployment.**

Keywords; Big Data; Data Locality; Data Locality Raito; MapReduce; Simulation; Virtual Envirinment; WorkflowSim

I. INTRODUCTION

MapReduce is originally invented by Google [1], and is widely used in the Big Data community due its high scalability and performance in processing large-scale datasets. It gains further popularity with its introduction of the open-source implementation, Hadoop [2]. It is de facto a popular programming model used in industry and many research groups to run their data-intensive applications.

The key concepts of MapReduce framework are the ability to divide data into a small size and send computation task to the data rather than data to computation task. The Task Scheduler in MapReduce is responsible to assign map task to a node containing its input data. If the scheduler fails to assign map tasks to nodes with input data, penalty time will occur for moving the data to the assigned node. When this happens, the performance of MapReduce will consequently degrade.

Data locality in MapReduce refers to the distance between of a map task or reduce task and the input data to be executed. In a physical machine cluster, data locality can be classified into node locality and rack locality [3]. There are more and more MapReduce deployment recently has recently in virtual environment [4, 5]. With the advancement of virtualization technology, it is possible for multiple deployments of nodes in MapReduce as virtual machines (VMs) on a single physical machine, called host. For better understanding of data locality in virtual MapReduce cluster node locality ought to be extended to both VM locality and host locality [4]. Previous works in [3], [5], [6], [7] and [8] have studied the data locality in MapReduce and proposed the scheduling algorithms for improving the data locality. However, most of these works showed the performance of data locality in term of overall percentage of locality, transfer data cost, execution time, and energy consumption. These measurements fail to provide more details of data locality such as percentage of each locality model.

As MapReduce architecture changing to virtual cluster, there is a lack of tool or measurement showing detail of data locality on such environment. This work proposes a novel measuring tool which can be used to investigate data locality ratio in MapReduce deployment on the virtual environment. The adopts a simulation based approach where WorkflowSim simulator [9] is utilized. Three popular scheduling algorithms were selected to this study, these are FIFO [5, 10, 11], Matchmaking [6] and Delay scheduling [3]. Data locality ratios of these three algorithms were compared which revealed useful information. The simulation approach in this work can be implemented to draw useful conclusion prior to actual physical deployment.

The organization of the paper is as follows. It begins with a brief description of MapReduce workflow and data locality model is elaborated in Section II. Related work is discussed in Section III. Section IV describes the implementation and extension to the WorkflowSim. Section V describes the simulation where results and relevant contribution are discussed in Section VI. The paper is concluded in Section VII where future work is also suggested.

II. MAPREDUCE AND THE DATA LOCALITY MODEL

In this Section, MapReduce framework and its workflow are described. The Data Locality Model is also explained here.

978-1-5090-4834-2/17/$31.00

A. *MapReduce Workflow*

In map task, data is processed according to given commands to produce an intermediate result. The reduce task then computes and summarizes these intermediate results to generate the final result.

MapReduce framework was introduced by Google in 2004 for processing large-scale datasets in a distributed and parallel platform. Datasets are assumed to be independently distributed across many nodes in a cluster and processed in a parallel manner. Fig. 1 depicts an example of the MapReduce workflow, it is initiated by dividing input data into collection of equally-sized chunks which are stored as multiple replicas on different nodes. Computation tasks are then assigned to process these data chunks. MapReduce comprises two computation tasks, *map task* and *reduce task*. In map task, data is processed according to given commands to produce an intermediate result. The reduce task then computes and summarizes these intermediate results to generate the final result.

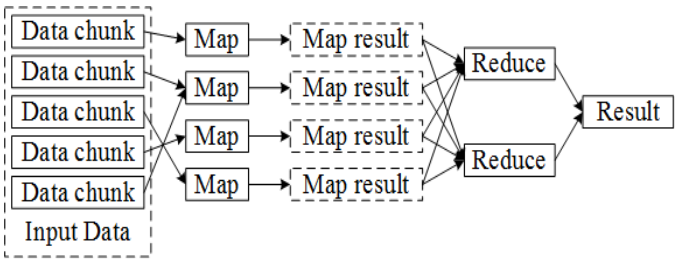


Figure 1. MapReduce workflow

B. *Data Locality Model*

In general, MapReduce cluster can be considered as a multi-level hierarchical network architecture [1, 3, 8, 12]. Physical machines are grouped in a rack server and communication among them is carried out via a rack switch. In higher level, rack servers are connected together by cluster switch or aggregate switch. Therefore, in physical MapReduce cluster, data locality model consists of Node locality and Rack locality. In virtual cluster, however, as a single physical machine, which can act as a host, is capable of dealing with multiple VMs for sharing resources. Node locality can be considered into two perspectives, these VM locality and Host locality.

In virtual environment, data locality of virtual MapReduce cluster, as described above, may lead to four scenarios as depicted in Fig. 2. These are :

1. *VM locality* : This is the scenario in Fig. 2(a)
2. *Host Locality* : This is the scenario in Fig. 2(b)
3. *Rack locality* : This is the scenario in Fig. 2(c)
4. *Off-rack locality* : This is the scenario in Fig. 2(d)

where communication occurs beyond a physical rack.

In practice, combination of these four scenarios usually occurs. Nevertheless, among these scenarios, VM locality is ideal as map task is assigned to VM node containing its input data, called *local task*. This scenario prevents communication task of VM locality, as extra communication task can occur otherwise. Hence, Off-rack locality ought to be avoided as much as possible.

Referring to Fig. 2, the work in [13] is the first to introduce the communication cost model in MapReduce cluster among these four localities. As no extra communication task occurs in VM locality, therefore the communication cost can be considered as zero. It was concluded that if communication cost of physical node (i.e. Host locality) is equal to c, then the communication cost of Rack locality and Off-lack locality, would be $4c$ and $6c$ respectively. Note that, while communication latency may be determined from the model suggested in [13], it is an overall value where ratios among the four scenarios may not be apparent.

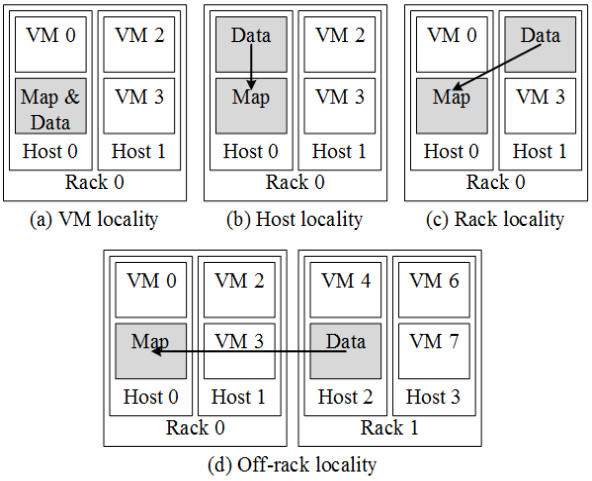


Figure 2. Data locality virtual MapReduce cluster

III. RELATED WORK

There have been several studies on data locality in MapReduce [3 – 7] and [13] from various aspects that may affect the performance. This work is concerned with two aspects in data locality, the scheduling and the evaluation. Hence, this Section discusses previous works which proposed scheduling algorithms for improving and those which presented tool for evaluating data locality.

In the Hadoop framework for MapReduce implementation, FIFO is the well known default algorithm to process incoming tasks by their order of arrival [5], [10] and [11]. As FIFO performance did not always yield satisfactory performance for data locality, Improvement was carried out in several works such as [3], [5], [6], [7] and [8] by proposing scheduling algorithms. Nevertheless, a common conclusion that can be drawn among them is that that assigning map task in MapReduce has the most impact to data locality. Match making algorithm [6] was proposed which attempted to search a compute node with current task's input data first. If the current task is not found, a free node is assigned to. Similar strategy was proposed in Delay Scheduling algorithm [3], but instead of using a free node, it leaves current task in a queue and carry on to another task. Delayed task is allowed to wait for a node with its input data within predefined time slot. The advantage of multithreading architecture is utilized in

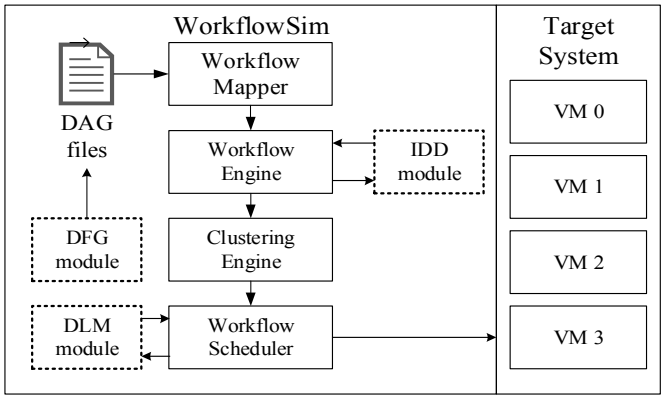
Multithreading Locality Scheduler (MLT) [5] by performing multithreading scheduling on a cluster of nodes resulting in a sub-block in the MapReduce cluster. Task scheduling of each block is handled by thread finding a node in a sub-block with input data for improving data locality. Some algorithms use data prefetching approach in MapReduce [7, 8]. Data prefetching could help to avoid penalty time by moving data from a node storing input data to a node which is executed before map task arrives. High Performance Scheduling Optimizer (HPSO) [7] also applied data prefetching technique. It predicts the most appropriate node to future map task and prefetch data to a node which executes future map task. The work in [8] combines scheduling and routing algorithm into a Joint Scheduler. It maintains data locality and network load balance in order to avoid network congestion during data prefetching phase.

There are relatively few works which focus on data locality evaluation. To evaluate the performance of proposed scheduling algorithms, previous works presented the performance in terms of overall percentage of locality, execution time, and energy consumption. However, these measurements do not reflect to the characteristics of data locality directly. LocalitySim [13] attempted to evaluate data locality in MapReduce, it presented data locality in term of communication cost of each locality model. This has a main drawback that users have to interpret data locality among different scenarios within MapReduce themselves.

Data locality ratio among the four different scenarios, as described in Section II, which can happen in MapReduce in virtual environment, will enable user to understand the performance of MapReduce better. Hence, this work attempts to implement modules within WorkflowSim which enables evaluation of MapReduce performance by means of data locality ratio.

IV. THE EXTENSION OF WORKFLOWSIM

In testing and evaluation of distributed computer systems, simulation is one such powerful method due to its practicality and flexibility without having to configure real physical deployment. Simulation of distributed computer systems in virtual environments is adopted in this work too, it is carried out by means of WorkflowSim. It is an extension of CloudSim simulator [14] and is currently a popular simulation tool for such environment. In WorkflowSim, Directed Acyclic Graph (DAG) model is used to represent the workflow model. In DAG model, a node represents computation task, while a directed edge represents dependency among tasks. DAG model is described in XML format which is currently defined by the XML schema of the workflow management system known as Pegasus WMS [15]. Fig. 3 depicts four major modules used to simulate target application in WorkflowSim.



DFG - DAG File Generator
IDD - Input Data Distributor
DLM - Data Locality Measurement

Figure 3. Modules in WorkflowSim

Once simulation is initiated, Workflow Mapper imports the workflows of target application in DAG files and maps them as a list of tasks to send them to the execution engine. Workflow Engine then generates virtualized infrastructure as user defined (i.e. virtual machines and physical host). It also initiates task scheduler, known as Workflow Scheduler. This module is responsible for assigning a task to a target system, depending on the predefined scheduling algorithm. It is possible for a user to implement and test a scheduling algorithm in Workflow Scheduler. Clustering Engine is situated between Workflow Engine and Workflow Scheduler. It merges similar tasks into a group in order to decrease scheduling overhead. It is possible for a user to disable Clustering Engine if native workflow is preferred.

As WorkflowSim uses DAG model to control workflow, MapReduce Workflow in DAG model and the extension to WorkflowSim in this work are described as follows :

A. MapReduce Workflow in DAG Model

As described in Section II, MapReduce workflow may be mapped into DAG model for ease of understanding, therefore the workflow in Fig. 1 may be transformed into DAG model as shown in Fig. 4.

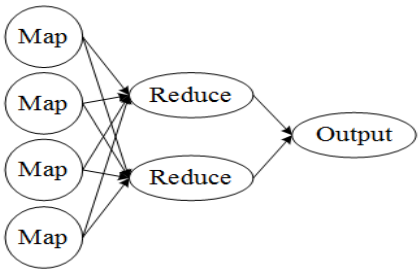


Figure 4. MapReduce workflow in DAG model

As Map tasks are independent and their execution can start at anytime when VM nodes are available, while reduce tasks are dependent which have to wait until all relative map tasks are terminated, this enables WorkflowSim to simulate MapReduce workflow appropriately.

B. *Scope and Asssumptions*

In practice, it may be possible for map tasks and reduce tasks to happen for several cycles before an output is obtained in MapReduce application. In order to simplify this study, the workflow is scoped that an output is obtained after a single cycle of map tasks followed by reduce tasks. Also map tasks scheduling is of main concern in this work, therefore, the following four conditions is assumed : (1) The number of map tasks is equal to that of data chunks, (2) A map task processes only one specific data chunk, (3) Reduce tasks may commence after all map tasks are computed, and (4) The data locality is assessed during tasks scheduling of map tasks.

It is worth noting that the result of this study is still valid within this scope and assumptions.

C. *Implementation*

This study proposes a novel measure of data locality called 'data locality ratio' in MapReduce application by means of simulation using WorkflowSim. The work extends the WorkflowSim by implementing three additional modules, these are *DAG File Generator (DFG)*, *Input Data Distributor (IDD)*, and *Data Locality Measurement (DLM)*. These modules correspond to the three dash line boxes in Fig. 3.

1) *DAG File Generator (DFG)*

DFG is responsible for generating a DAG file of MapReduce workflow. The DAG file format follows the Pegasus XML schema. User is able to define the number of map and reduce tasks and the input data size with its run time. Fig. 5(a) shows an example of a DFG output file. Note that the job element in XML represents map task or reduce task.

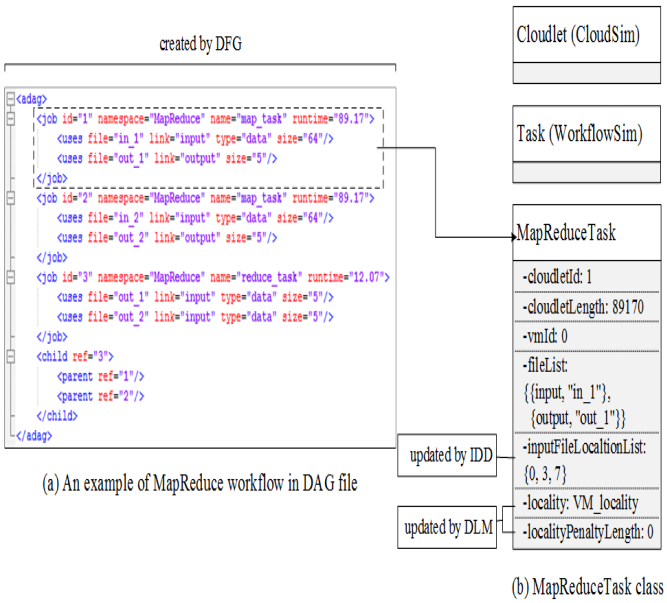


Figure 5. Information flow among DFG, IDD and DLM modules in WorkflowSim

The example DFG file comprises two map tasks in the first two blocks and one reduce task in the block which follows. The 3rd block (reduce_task) block can be considered as a child of the first two map task blocks indicating dependency between map task and reduce task by defining the parent-child relationship. This can commence only after its two parents are completed. The DGF output file is fed to Workflow Mapper for generating a list of tasks. During its generation, the information of task in DAG output file (e.g. id, runtime, input file name and output file name) is added to an instance of task. This is then sent to Workflow Engine as indicated in Fig. 3.

2) *Input Data Distributor (IDD)*

MapReduce processes large datasets by dividing them into small chunk. Each chunk will be replicated and stored across cluster. IDD is implemented to support to replicate and distribute each data chunk over VM nodes. It allows user defining the number of replication. As mentioned in our assumptions, an output is obtained after a single cycle of map tasks followed by reduce tasks, therefore the map tasks represent unique data chunks as well.

In order to simulate the replication and distribution of data chunks, *MapReduceTask class* is implemented as an extension of *Task class* of WorkflowSim and *Cloudlet class* of CloudSim as shown in Fig. 5(b). *MapReduceTask* is designed to record locations of data chunks represented by *inputFileLocationList field*. Referring to Fig. 3, after Workflow Mapper receives a list of tasks, Work Engine creates VM nodes and hosts in clusters and initiates its scheduler (i.e. Workflow Scheduler). The task of IDD is to fetch this list of tasks and VM nodes, then it replicates and distributes input data chunks and records their locations by noting their VM id of data chunks in *inputFileLocationList*. After this is complete, IDD returns a list of tasks with their locations of input data chunks back to Workflow Engine.

3) *Data Locality Measurement (DLM)*

Referring to Fig. 3, DLM is implemented to determine data locality of tasks by working with Workflow Scheduler. Once a list of tasks and VM nodes are created, Workflow Engine begins by passing unassigned tasks to Workflow Scheduler, where tasks are assigned to VM nodes. The assigning policy depends on scheduling algorithm defined by user.

In each scheduling cycle, scheduler assigns unassigned tasks to available VM nodes. Referring to Fig. 5 (b), task which is successfully scheduled is marked by VM id in *vmId field* and changes its status to scheduled task. Prior to submission of scheduled tasks to VM nodes, DLM intercepts Workflow Scheduler and to determine data locality model of each task by comparing VM id in *vmId field* to VM id in *inputFileLocationList field*. The result is recorded in *locality field*. Penalty time for moving data occurs if it is not VM locality. DLM then updates *localityPenaltyLength field* by the user defined penalty time model. The penalty time is added in task's run time when task is executed in VM node. As DLM accumulates the number of map tasks and the number of data localities during simulation. Once the simulation is complete, values for all four types of data locality are available in DLM. This enables DLM to determine 'data locality ratio'. This ratio represents the ratio of 'VM Locality : Host Locality : Rack Locality : Off-rack Locality'. This module is implemented with an option of displaying this 'data locality ratio'.

V. SIMULATION

As stated on Section I, three scheduling algorithms were selected for this study, there are of FIFO, Matchmaking, and Delay Scheduling. As data prefetching has no effect on them, this allows the study to focus on the impact of scheduling algorithms on data locality ratio directly. All three scheduling algorithms were implemented in this study. FIFO was implemented in its original concept where map tasks are dealt with in their order, this is to be referred to as 'FIFO'. Two versions of Matchmaking and Delay Scheduling were implemented. Their original versions as described in [6] and [3] as a modified version of each scheduling. The rationale for the modified versions is that the communication task of Host locality may be acceptable in certain architecture of virtual MapReduce cluster. Simulation in this study is intended demonstrate an opportunity to adjust scheduler configuration and to investigate options to improve data locality.

In its original versions of both Matchmaking, and Delay Scheduling, the scheduler attempts to assign a map task to VM node with the best VM locality. If this fails, the task is then assigned any VM node available. For brevity, these two original versions will be referred to as 'MM_VM' and 'DL_VM' respectively. As Host locality is preferable to Rack locality and Off-rack locality, in the two modified versions, the scheduler is more flexible. If it fails to assign a VM node with the best VM locality, the task is assigned to a VM node with Host locality first, if possible, before assigning any VM node available. For brevity, these two original versions will be referred to as 'MM_HOST' and 'DL_HOST' respectively. Therefore, this study simulated five task schedulers altogether (i.e. FIFO, MM_VM, MM_HOST, DL_MM and DL_HOST) to measure their data locality ratio and execution time.

Simulation is carried out on virtual MapReduce cluster consisting of 16 virtual machines of 8 physical hosts populated in 4 rack servers where each rack contains two physical hosts. Fig. 6 depicts the configuration of the MapReduce cluster.

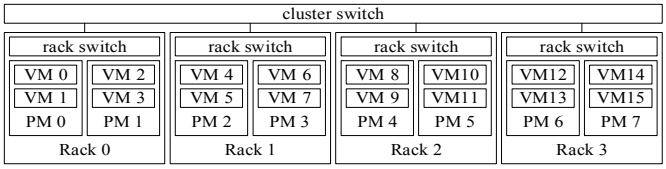


Figure 6. The virtual MapReduce cluster

Table 1 shows the parameters for the virtual machine and the physical host.

TABLE I. VM AND HOST PARAMETERS

Parameters	VM	Host
Number of CPUs	1	2
MIPS per CPU	1,000	2,000
Ram (MB)	512	2048
Storage (MB)	10,000	1,000,000
Bandwitdh (Mbps)	1,000	10,000

In the simulation, each scheduling algorithm is simulated over the MapReduce application with 1,000, 2,500, 5,000, and 7,500 map tasks which are approximately 60, 156, 312, and 469 GB of input data size respectively. The input data is divided into a chunk size of 64 MB and each chunk has three replicas as used in the default configuration of Hadoop framework [16]. The communication cost model proposed in [13] is adopted in the simulation. For 1,000 Mbps bandwidth of VM node, the transmission time for 64-MB data chunk between node within the same host is 488 milliseconds (ms). Therefore, the transmission time for sending data chunk among nodes within the same rack is 1,952 ms and 2,928 ms in case of between racks.

VI. RESULTS AND DISCUSSIONS

As the objective of this study is to present a new measure of the impact of scheduling algorithms on data locality. Fig. 7 reveals the data locality ratio of each scheduler processing over four data sizes.

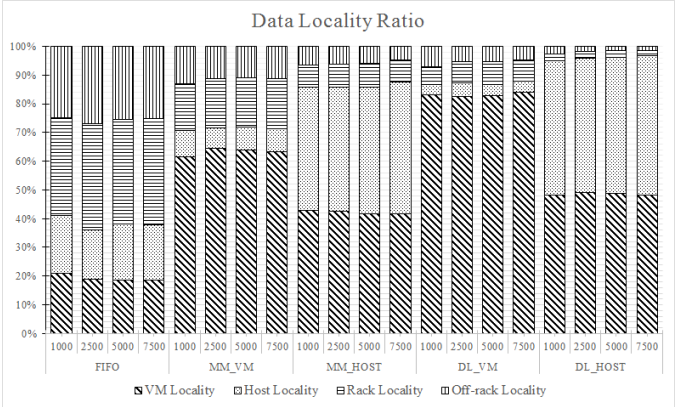


Figure 7. Data locality ratio of all five scheduling alforithms

As mentioned in Section II, VM locality is the most preferable. Referring to Fig. 7, DL_VM algorithm yielded the best of around 83% of VM locality in all data sizes while FIFO yielded the worst VM locality. This result is in accordance with the work in [6] where Delay Scheduling achieved almost 100% of node locality. Fig. 8 reveals the execution time of all versions simulated.

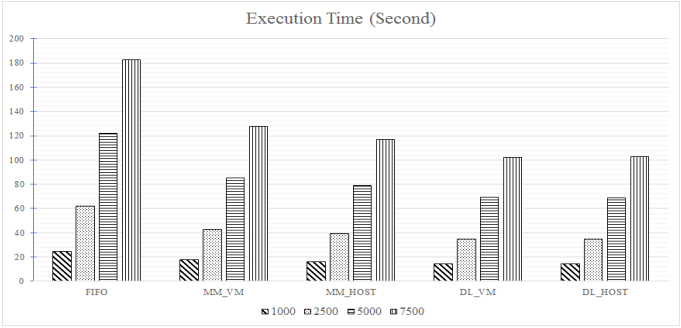


Figure 8. Execution Time of all five scheduling algorithms

Referring to Fig. 8, DL_VM and DL_HOST Delay scheduling, yielded faster execution time than others. Taking both results in Fig. 7 and Fig. 8 into consideration together, it can be concluded that while DL_HOST achieved lesser VM locality than DL_VM, DL_HOST compensates this with better the ration among Host locality to Rack Locality to Off-rack locality. Their difference in execution time is negligible. The similar can be said in comparison between MM_VM and MM_HOST. Note that the results in this study reflect the circumstance where high ratio of 'VM locality to Host locality to Rack to Off-rack locality' is preferred. In other circumstance, the penalty time of Host locality may have less impact on execution time, while the penalty times of Rack and Off-rack locality may have significant impacts. On the other had, in circumstance such as rack server with 10-Gigabit Ethernet switch, the penalty time of Rack locality may be acceptable while in low-performance physical host, the penalty time of Host locality may not.

The objective of this study is not about determining which combination of configuration and scheduling algorithm yields the best VM locality and execution time, as there are many variables to consider and beyond the scope of this study. Apart from studying the five selected scheduling algorithms, the main objective of this study is to present a new measurement, 'data locality ratio', of MapReduce application in virtual environment. This measure can provide crucial information for improving data locality and the performance of MapReduce application as it offers an efficient way for selecting and tuning scheduler and system configurations suitable for an environment prior to its actual physical deployment.

VII. Conclusions and Future work

Data locality is the critical factor of MapReduce performance. Numerous previous works have focused on improving data locality by proposing different kind of scheduling algorithms as alternatives to FIFO algorithm, a default algorithm in Hadoop. Nevertheless, performances were presented in term of overall percentage of locality and the execution time. However, knowing the ratio of each type of data locality is useful information for better understanding the performance of a scheduling algorithm. Especially in MapReduce performance in virtual cluster, this can be translated into knowing the ratio 'VM locality : Host locality : Rack locality to Off-rack locality'.

This study proposes a novel measurement for MapReduce performance in virtual cluster called 'data locality ratio'. It expresses the ratio among these four types of locality mentioned above. Simulation was carried out on virtual MapReduce cluster with various realistic configurations. The 'data locality ratio' is determined by extending the WorkflowSim simulation to include three implemented modules. The results revealed that the original Delay Scheduling achieved high data locality ratio. The approach in this study enables users to obtain crucial information to select and tune scheduler and system configurations prior to its actual physical deployment.

Future studies can be extended to MapReduce cluster in heterogeneous distributed system e.g. physical cluster, virtual cluster, and cloud computing. Data security and privacy constrains imply that data locality constrain and reliable processing on heterogeneous systems required further investigation. It is anticipated that new generation of workflow tool and scheduler have to be designed in order to meet these challenges.

Acnknowledgement

The authors are grateful to School of Information Technology (SIT), King Mongkut's University of technology Thonburi (KMUTT) for the scholarship of Mr. Peerasak Wangsom and the computing facilities throughout this research.

References

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107-113, 2008.

[2] *Apache Hadoop*. Available: http://hadoop.apache.org/

[3] Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. EuroSys'10*, 2010, pp. 265-278.

[4] X. Ma, X. Fan, J. Liu, H. Jiang, and K. Peng, "vLocality: Revisiting Data Locality for MapReduce in Virtualized Clouds," *IEEE Network*, vol. 31, pp. 28-35, 2017.

[5] Q. Althebyan, Y. Jararweh, Q. Yaseen, O. AlQudah, and M. Al-Ayyoub, "Evaluating map reduce tasks scheduling algorithms over cloud computing infrastructure," *Concurr. Comput. : Pract. Exper.*, vol. 27, pp. 5686-5699, 2015.

[6] C. He, Y. Lu, and D. Swanson, "Matchmaking: A New MapReduce Scheduling Technique," in *Proc. CLOUDCOM'11*, 2011, pp. 40-47.

[7] M. Sun, H. Zhuang, C. Li, K. Lu, and X. Zhou, "Scheduling algorithm based on prefetching in MapReduce clusters," *Applied Soft Computing*, vol. 38, pp. 1109-1118, 1/2016.

[8] W. Wang and L. Ying, "Data locality in MapReduce: A network perspective," in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2014, pp. 1110-1117.

[9] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. E-SCIENCE' 12*, 2012, pp. 1-8.

[10] L. Thomas and R. Syama, "Survey on MapReduce Scheduling Algorithms," *International Journal of Computer Applications*, vol. 95, pp. 9-13, 2014.

[11] T. White, *Hadoop: The Definitive Guide*. Sebastopal, CA: O'Reilly Media, 2015.

[12] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. SIGCOMM'08*, 2008, pp. 63-74.

[13] A. H. Abase, M. H. Khafagy, and F. A. Omara, "Locality Sim: Cloud Simulator with Data Locality," *CoRR*, vol. abs/1701.01648, 2017.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, S. A. F. D. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, pp. 23-50, 2011.

[15] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, et al., "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Sci. Program.*, vol. 13, pp. 219-237, 2005.

[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *Proc. MSST'10*, 2010, pp. 1-10.