Optimal MapReduce Job Scheduling algorithm across Cloud Federation

Thouraya Gouasmi[1], **Wajdi Louati**[2] **and Ahmed Hadj Kacem**[1]
[1]ReDCAD Lab, Faculty of Economics and Management, University of Sfax, Tunisia
[2]ReDCAD Lab, National Engineering School, University of Sfax, Tunisia

Abstract— *Processing data-intensive applications with MapReduce in geo-distributed cloud federation has emerged recently with several challenges. Distributing the overhead Map tasks and re-optimizing virtual machines in cloud while taking advantage of data locality represents today a key for improving job performance and guarantee security and privacy. This paper proposes an exact scheduling algorithm to process MapReduce data-intensive applications across geo-distributed clusters. The proposed algorithm takes advantage of data locality while reducing both VM cost and data transfer cost (between clusters) subject to Deadline. The goal is also to provide a baseline for benchmarking and evaluation of future heuristic scheduling algorithms in the literature. The proposed exact scheduling model is formulated and solved as a mixed integer program. Performance of the proposed algorithm is reported.*

Keywords: MapReduce; Cloud Federation; Distributed MapReduce Scheduling; Linear Programming;
PDPTA′17 : LATE BREAKING PAPER;

1. Introduction

In the Data era, there is a "big data" revolution in which the rapid and continues increase of huge data has been exploding. Today, MapReduce [1] has become an important distributed processing model for large-scale data-intensive applications like data mining, web indexing and web search. Its open source software framework Hadoop [2] [3] [4] is a successful implementation for distributed storage and processing of very large data sets on computer.
Deploying MapReduce on top of cloud computing [5] can take the advantage of data locality (assigning tasks to nodes that contain their input data)[6]. There has been much recent work, like [7–17], on provisioning distributed MapReduce jobs across federated clouds (or Multi-clouds) [18]. These contributions mainly address the scale-out model and scheduling issues with various concerns placed on clusters capabilities, energy reduction, data transfer time, data privacy, data locality and network performance. Two main approaches have been identified for processing MapReduce on top of federated clouds.
First, some works mainly focused on splitting the data set and distribute them across multiple MapReduce clusters, and balances the workload by assigning tasks in accordance to some properties like the capabilities of cluster and so on [8–10], [13], [14], [16]. This approach takes advantage of MapReduce parallel processing, fulfill several users requests by external computational and storage resources, while satisfying requirements and SLA objectives. However, this approach has critical issues. Some data must keep privacy and cannot be burst across distributed domains/clusters or aggregated for centralized analyzing. Yet, for large-scale data, this procedure is extremely time-intensive and complex due to limited network bandwidth and network cost traffic.

In the second approach, some works investigated in highly geo-distributed data processing. All data are gathered from multiple datacenters (or clouds) to a centralized one for job computation. This approach is neither efficient nor practical as the volume of large-scale data grows exponentially. Moreover, some data are restricted to certain datacenters due to security and privacy constraints (like log files and mails). Moving very large data becomes prohibitive in terms of bandwidth cost and processing time. Instead, a recent trend is to distribute MapReduce computation across geo-distributed multi-clusters where the data sets are located (i.e. data locality), thus avoiding data movement and reducing the costs (e.g. bandwidth, time, etc.) [7], [11], [12], [15], [17]. Despite that these proposals already improved performance, they are still however relying on centralized cloud managers or entities responsible for scheduling, orchestrating and coordinating jobs and data across distributed clusters. For large-scale data, using a centralized entity to gather aggregated results is costly and time intensive.

To the best of our knowledge, no work has proposed an exact scheduling algorithms of MapReduce job across cloud federation while reducing the cost (using MAP task outsourcing across the cloud federation) subject to Deadline constraint. This paper proposes an optimal scheduling solution for mapreduce scheduling.
The rest of the paper is organized as follows. Section 2 outlines related work. Section 3 describes our architecture. Section 4 describes the problem definition and exact model. Section 5 presents the performance evaluation of our approach. Finally, Section 6 concludes the paper.

2. Related Work

Several proposals have been put forward (like [7], [11], [12], [15], [17]) to distribute MapReduce computation across geo-distributed multi-clusters where the data sets are located while reducing the costs.

For instance, in [11], Zhang et al. have compared the single-datacenter Hadoop deployment with multiple datacenter Hadoop deployment to identify the performance issues inherent in a geographically distributed clouds. Authors proposed some system-level optimizations for performance improvement, including prediction-based job localization, data prefetching and configurable HDFS data placement. The work does not provide models and algorithms for the proposal.
In [12], authors formalize a centralized optimization of MapReduce computation over multi-cloud environment, thus a non-linear integer programming model is defined for reducing VM cost subject to deadline constraint. The work relies however on a centralized optimizer to gather results.
To guarantee non-violence of sensitive data, an approach called Fed-MR framework [7] is proposed. The MapReduce program is separated as several map and reduce tasks inside the job submission flow and scheduled to clusters where the sets of data are located. Fed-MR introduced a proxy based expansion to the original MapReduce workflow so that a single MapReduce job can be transparently executed across multi-clusters.
The Fed-MR proposed a new hierarchical framework to keep privacy of sensitive data but does not offer algorithms of MapReduce scheduling in distributed clusters. Fed-MR relies on a centralized entity called Top Cloud responsible for job coordination between Region Clouds.

To summarize, existing works propose heuristic scheduling algorithms, tried to relax some constraints and objectives. The optimization objectives mentioned above have been considered separately. The main contribution of our work is to propose an exact optimization for mapreduce job schedulin that takes jointly into account cost, data transfer, and deadline.

3. Exact Scheduling Architecture

Figure 1 depicts the general framework to distribute Map tasks over geo-distributed clusters in cloud Federation. These clusters are managed by different cloud providers. We assume that the MapReduce data input are already stored in the local disks of clusters. Figure 1 shows three clusters (Cloud A, Cloud B and Cloud C)interconnected via high speed networks.
Upon receiving a user job, one cluster (e.g. cloud A) divides it into sub-jobs and distributes them across *R* neigboring clusters of the Federation (e.g. cloudB, cloudC). In each Cluster, Hadoop divides the input data into splits according

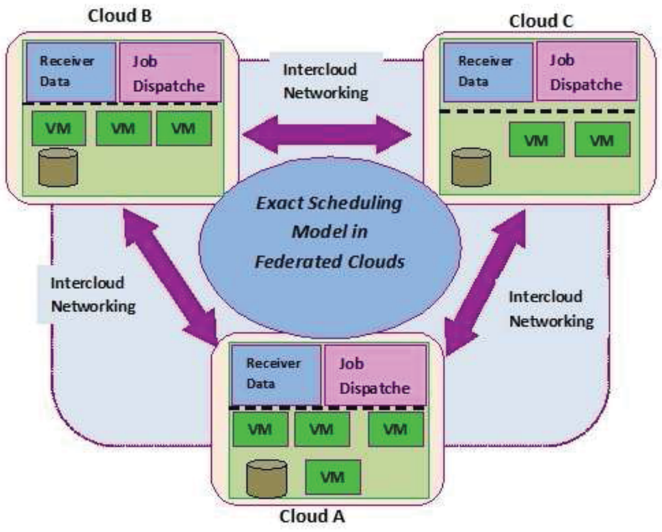

Fig. 1: MapReduce scheduling architecture for MAP tasks outsourcing across cloud federation

to the HDFS block size and also divides the sub-job into Map tasks running in slots. In this work, the slot represents a Virtual Machine instance running one Map task. Each Map task is associated to one data split.
In the conventional approach, when the number of available VMs in a given Hadoop Cluster is less than the number of MAP tasks, new VMs must be allocated. To obtain the maximum benefit from Cloud federation and since the VM cost could be different from one cloud provider to another, outsourcing the MAP Tasks and their associated data splits to idle VMs located in other Clusters (rather than allocating new instances) may reduce the cost. Obviously, the cost of migrating tasks and data to other clusters (including bandwidth cost and VM allocation cost) should be less than the cost of instantiating in-house VMs to compute job tasks. This is exactly the motivation of this work whose goal is to ensure cost-efficient scheduling of MapReduce tasks in the federation.
Figure 1 present our exact scheduling model between three cloud interconnecting with higher network. Each cluster in the federation includes two components: Receiver data and Job Dispatcher. The Receiver data entity consist to received Map tasks and data from their neigboring clusters of the Federation and the Job dispatcher information about the cluster (idle VMs, VM cost,...) and dispatch their Map tasks and data to cloud provides lower VM and bandwidth cost in a defined delay.

4. Problem Modeling

4.1 Map tasks scheduling problem formulation

As depicted in Figure 1, in a given cluster, each Map task is running in one VM (slot) and associated to one data split.

Each cluster r_i has N_{r_i} Map tasks (which represents also the number of data splits).
Let S_{r_i} denote the number of slots used to run the map tasks. In the case where the number of tasks is greater than the number of slots ($N_{r_i} > S_{r_i}$) in a given cluster, we propose to outsource the overhead Map tasks as well as their split data to another cluster. The number of Map tasks that exceeds the number of VMs of one Cluster r_i can be presented as:

$$M_{r_i} = N_{r_i} - S_{r_i} \ With \ i : 1 \cdots R \tag{1}$$

In the case where the number of tasks is less than the number of slots ($N_{r_i} < S_{r_i}$), some slots become inactive (idle). Let N_{r_i} denote the number of idle slots in the cluster r_i not running Map tasks:

$$A_{r_i} = S_{r_i} - N_{r_i} \ With \ i : 1 \cdots R \tag{2}$$

In this paper and for simplicity, we consider that all VMs (i.e. slots) in all clusters have similar virtual hardware configuration (or template, e.g. small). Since the VM cost is different from one cloud provider to another, the cost of a VM instance (as defined in [19]) on one cluster r_i can be presented by ($Perf^{r_i}$, Px^{r_i}) where $Perf^{r_i}$ denotes the performance of the virtual machine in million instruction per second (MIPS), and Px^{r_i} denotes its price per time unit (e.g. $ per Hour). Since each Map task consists of a number of Instructions *I*, the cost of one VM instance C_{r_i} can be measured as:

$$C_{r_i} = I/Perf^{r_i} \times Px^{r_i} \tag{3}$$

The cost of instantiating new VMs in a cluster r_i required to perform M_{r_i} Map tasks can be measured as:

$$C_{M_{r_i}} = \sum_{k=1}^{M_{r_i}} C_{r_i} \tag{4}$$

In the case of Map task outsourcing, the data splits, associated to the tasks, must be also moved. In the ideal case when $M_{r_i} = A_{r_i}$, to outsource data splits (M_{r_i}) from one cloud to another, the allocation cost of network bandwidth must be considered. Each cloud provider fixes prices for the inbound (i.e. data going into the Cloud) and for the outbound (i.e. data going out of the Cloud) data transfers in $ per GB. Let $Cl_{r_ir_j}$ denote the cost of allocating network bandwidth to transfer M_{r_i} data splits from one cluster r_i to another r_j and $P_{r_ir_j}$ presents their associated allocation price (in $ per GB).

$$Cl_{r_ir_j}^{M_{r_i}} = M_{r_i} \times D_{r_i} \times P_{r_ir_j} \tag{5}$$

where D_{r_i} denotes the block size of one data split. Consequently, the cost of allocating VMs from another cluster r_j required to perform Map tasks M_{r_i} can be measured as:

$$C_{M_{r_i}} = \sum_{k=1}^{M_{r_j}} C_{r_j} + Cl_{r_ir_j}^{M_{r_i}} \ where \ j \neq i \tag{6}$$

The time required to transform M_{r_i} data splits from one cluster r_i to a cluster r_j is denoted by $T_{l \ r_ir_j}$ and express by the amount of data to be migrated on available bandwidth $Bn_{r_ir_j}$(in GBs). The Map phase of a sub-job consists of multiple iterations (waves) according to the number of Map tasks and Map slots together. Therefore, the time consumed by each cluster, denoted by T_{MP}, to run a sub-job is the execution time of one map task T_M multiplied by the number of waves. Therefore, the execution time of sub-job can be presented as follow [20]:

$$T_{MP} = \lceil \frac{N_{r_i}}{S_{r_i}} \rceil \times T_M \tag{7}$$

The execution time of one map task in a VM is given by:

$$T_M = I/Perf^{r_i} \tag{8}$$

The job completion time, denoted by T_C, is determined by its last completed sub-jobs across the clusters.

$$T_C = \max_{i=1 \cdots R} T_{MP_{r_i}} \tag{9}$$

Therefore, finishing a portion of the job quickly at one cluster does not necessarily result in faster overall job completion time.
Let $T_{response}$ denote the entire job response time, defined as all time required to run the overall job MapReduce and is measured as:

$$T_{response} = T_C + \max_{i,j=1 \cdots R} T_{r_ir_j} \tag{10}$$

In fact, an optimal scheduling approach must meet some constraints. For instance, the job response time must not exceed a deadline D. The response time can be consequently presented as:

$$T_{response} \leq D \tag{11}$$

In this work, we only consider the cost metric of MapReduce jobs. The job cost is defined as the cost of all sub-jobs running on local or remote clusters. It is affected by various system and workload parameters which are listed in Table 1.

4.2 Exact model for scheduling and outsourcing overhead Map tasks across cloud federation

Scheduling and outsourcing Map tasks should be efficient to deal with distributed MapReduce performance degradation in cloud federation. Heterogeneous scheduling should handle differences in computing power of virtual machines and network performance. In this section, an exact and optimal scheduling solution for the problem introduced in the previous section. Thus, we provide a multi-attribute model of the outsourcing problem that considers VM cost, bandwidth cost, availability and deadline constraints. For this purpose, this problem is formulated as a MIP.

Table 1: Definition of Notation used in Modeling

Symbol	Definition
R	Set of clusters
N_{r_i}	Number of map tasks of sub-job at a cluster r_i
S_{r_i}	Number of slots of cluster r_i
M_{r_i}	Number of remaining Map tasks of one cluster r_i
A_{r_i}	Number of available Slots in one cluster r_i
C_{r_i}	Cost of one VM instance
$C_{M_{r_i}}$	Cost of M_{r_i} Map tasks on a cluster r_i
$Cl_{r_i r_j}^{M_{r_i}}$	Data transfer cost on the bandwidth path between a cluster r_i and a cluster r_j
C_R	Overall Job cost distributed into all clusters R
$T_{l\ r_i r_j}$	Transfer time of data between r_i and r_j
T_{MP}	Execution time of sub-job Map phase on a cluster r_i
T_C	Completion time of one sub-job
$T_{response}$	Response time of the overall job
T_M	Execution time of one Map task
$Bn_{r_i r_j}$	Available bandwidths between r_i and r_j
D	Deadline of job MapReduce specified by user
$x_{r_j}^{i' r_i}$	Binary variable equal to 1 when if cluster rj is hosted by Map task i' of remaining Map tasks M_{r_i}

4.2.1 MIP formulation

The objective is to efficiently outsource remaining Map tasks to VMs instances in the cloud federation while ensuring simultaneous reduced VM and bandwidth costs and respecting deadline. As finding the optimal solution to this kind of multi-attribute selection problems is NP-Hard [24], we propose a formulation using a linear integer program based on binary variable $x_{r_j}^{i' r_i}$. Variable $x_{r_j}^{i' r_i}$ is equal to 1 (resp. 0) if the cluster rj is hosted (resp. not allocated) by the Map task i' of the remaining Map tasks of the cluster r_i. The selected cluster r_j can be the source cluster r_i by instantiating new VMs.

Linear programming is a technique for the optimization of a linear objective function, subject to linear equality and inequality constraints [21]. The convenience and rich MIP formulation of model is presented in details in the next section.

4.2.2 Objective function

The objective function (12) of the linear program minimizes the MapReduce job cost while reducing the data transfer cost and the required VM cost supporting $x_{r_j}^{i' r_i}$ under some constraints (section 4.2.3).

$$\begin{aligned} min \sum_{r_j \in R} \sum_{i' \in M_{r_i}} \sum_{r_i \in R} x_{r_j}^{i' r_i} \times C_{r_j} \\ + \sum_{r_j \in R} \sum_{r_i \in R} x_{r_j}^{i' r_i} \times Cl_{r_i r_j} + \alpha \end{aligned} \tag{12}$$

where

$$\alpha = \sum_{r_i \in R} (N_{r_i} - M_{r_i}) * C_{r_i}; \tag{13}$$

Three main terms composed the objective function of MIP. The first term minimizes cost of outsourced Map tasks by reducing VM cost C_{r_j}. The second term minimizes bandwidth cost $Cl_{r_i r_j}$ of transfered Map tasks on a bandwidth capacity. A constant α is added to express cost of located Map tasks for each cloud r_i in order to complete the total job cost.

4.2.3 Constraints

a) Uniqueness constraint:

$$\sum_{r_j \in R\ r_i \neq r_j} x_{r_j}^{i' r_i} \leq 1, \quad \forall i' \in M_{r_i}, \quad \forall r_i \in R \tag{14}$$

Constraint 14 ensures that only one cloud r_j is selected for each Map tasks i'.

b) Availability constraint:

$$\sum_{i' \in M_{r_i}} \sum_{r_i \in R\ r_i \neq r_j} x_{r_j}^{i' r_i} \leq A_{r_i}, \quad \forall r_j \in R \tag{15}$$

Constraint 15 ensures that the requested slots(VMs) is allocated by all Map tasks are at most A_{r_j}.

c) Linearization related constraints:

$$\sum_{i' \in M_{r_i}} \sum_{r_j \in R} x_{r_j}^{i' r_i} = M_{r_i}, \quad \forall r_i \in R \tag{16}$$

Constraints 16 ensures coherence between Map tasks outsourced and the initial group Map tasks M_{r_i}, in which each Map tasks will be allocated to VM likewise of cloud r_i (when r_i provides VM cost C_{r_i} lower than the $C_{r_j} + Cl_{r_i r_j}$).

d) Bandwith time constraint:

$$T_{l r_i r_j} = \sum_{i' \in M_{r_i}} x_{r_j}^{i' r_i} / Bn_{r_i r_j}, \quad \forall r_i, r_j \in R \quad r_i \neq r_j \tag{17}$$

Constraints 17 expresses transfer time of data on available bandwidth network that should be different to null when the cloud r_i outsources Map tasks $x_{r_j}^{i' r_i}$ to another cloud r_j.

e) Deadline constraint:

$$\begin{aligned} \max_{r_i \in R} \max_{r_j \in R} \lceil \frac{N_{r_i} - \sum_{i' \in M_{r_i}} x_{r_j}^{i' r_i} + \sum_{i' \in M_{r_i}} x_{r_i}^{i' r_j}}{S_{r_i}} \rceil \times T_M \\ + \max_{r_i \in R} \max_{r_j \in R} T_{l r_i r_j} \leq D \end{aligned} \tag{18}$$

Constraint 18 guarantees that response time resulted after Map tasks outsourcing should not exceed a given deadline D.

f) Constraints and conditions:

$$Cl_{r_i r_j} = 0, \quad \forall r_i, r_j \in R, r_i = r_j \tag{19}$$

$$T_{lr_i r_j} = 0, \quad \forall r_i, r_j \in R, r_i = r_j \tag{20}$$

Constraints 19 and 21 define a condition on the time transfer and cost bandwidth of Map tasks between cloud r_i and r_j when selected cloud r_j is itself cloud r_i (no outsourcing).

g) Constraints for variable domains:

$$x_{r_j}^{i' r_i} \in \{0,1\}, \quad \forall r_i \in R \quad \forall r_j \in R \quad \forall i' \in M_{r_i} \tag{21}$$

5. Performance Evaluation

5.1 Evaluation Environments

The evaluation scenario involves a federation of clouds with different size (the number of clusters/Cloud datacenters varies from 2 to 10). The number of VMs per cluster is generated randomly in the [450, 5000] interval. As our model is data locality-aware, we assume that each cluster stored randomly input data size varies in Gbs from [10, 500] that is split among input blocks of 64Mb. Each Map task processes an input block data in one VM(slot). The pricing policy and computing power of a virtual machine and network configuration have been randomly generated by picking parameters according to values observed in real systems and logs of MapReduce applications.

The proposed exact algorithm is developed using AMPL [23] with the library ILOG of IBM ILOG CPLEX [22]. The experiments were carried out on an Intel Core i7, 3.4GHz, 8GB of RAM and running Linux Ubuntu-12.04. The cloud federation topologies are generated randomly using Matlab. The number of clusters within the federation should be in [2,12].

5.2 Evaluation Results

To evaluate the effectiveness of our proposed FedSCD algorithm, we compare its performance against a basic and heuristic conventional MapReduce scheduling algorithm (like Fed-MR [7]).

The evaluation scenario will be discussed related to two performance metrics (Job costs and Deadline).

5.2.1 Scenario 1

To analyze the behavior of the exact outsourcing model and its scalability with problem size, a first experiment is set up to evaluate the job costs generated on increasing input data size. The job cost is defined as the sum of costs of VMs (used for Map and Reduce Computation) and costs of bandwidth (related to MAP tasks migration between geo-distributed clusters). The number of clusters in the federation are fixed to both 8 and 10 clusters.

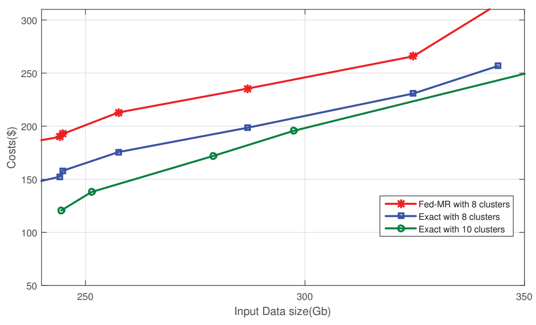

Fig. 2: Impact of different input data size on Job Cost.

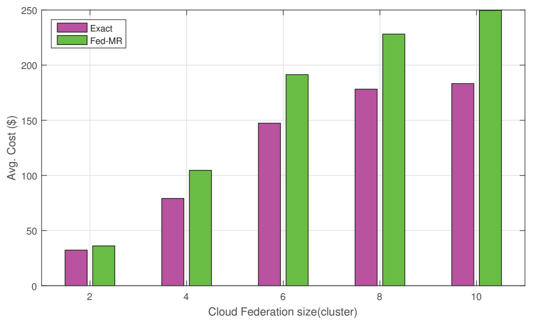

Fig. 3: Average MapReduce Job costs for different cloud federation size.

Figure 2 shows that the job costs generated by our exact scheduling algorithm is reduced than the costs incurred with the Fed-MR approach (in the case of 8 clusters) when the data input size increases (up to 350 GB). When the federation size increases (up to 10 clusters), the job cost decreases compared to the case of 8 clusters. The exact scheduling solution might be more relevant for large scale federation size.

5.2.2 Scenario 2

A second experiment consists in evaluating the impact of the exact model on the job cost when the federation size of clusters increases from (2 to 10). We generated 50 different MapReduce jobs and we computed an average cost for both exact and conventional algorithm (Fed-MR). As depicted in Figure 3, the exact algorithm reduces the cost by optimizing the number of unused (idle) VMs. The exact solution obtains a cost reduction of average 40% compared to the conventional algorithm while increasing the number of clusters and input data size.

5.2.3 Scenario 3

The aim of the third experiment is to study the effect of user-defined deadline on the job cost. 50 MapReduce jobs are sequentially generated with different sizes composed of

MAP tasks varying from 0 to 4000. As depicted in Figure 4, when the user-defined deadline increases, the cost decreases especially when the size of the federation grows from 2 to 10. This is explained by the fact that by increasing the deadline, the transfert time of data inter-cluster may increase which allows the exact algorithm to increase the number of outsourced MAP tasks. This induces a decrease in the cost.

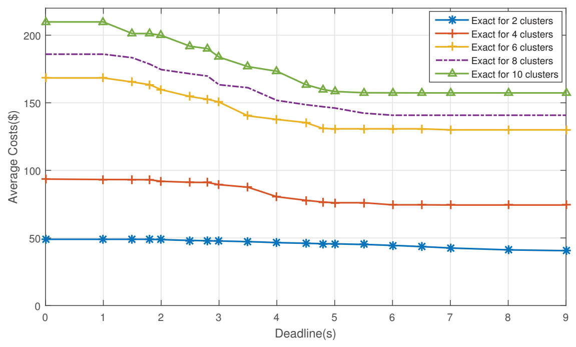

Fig. 4: Average MapReduce Job costs per Deadline.

6. Conclusion

In this paper, we propose an exact approach to process MapReduce jobs in geo-distributed cloud federation with goal of minimizing the communication (bandwidth) and VM costs respecting a user-defined deadline. To obtain an optimal solution, we formulated the exact scheduling problem as MIP formulation. The experimental results show that our proposed algorithm can gain better performance with lower MapReduce job cost while respecting a deadline. This works offer an exact solution in order to provide a baseline for benchmarking and evaluation of heuristic multi-objective heuristic scheduling algorithms. Future work will explore the use of the exact scheduling algorithm into some real-world applications.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *6th Symposium on Operating Systems Design and Implementation OSDI'04*, San Francisco, CA, 2004.
[2] Hadoop. [Online]. Available: http://hadoop.apache.org
[3] Hadoop Distributed File System. [Online]. Available: http://hadoop.apache.org/docs/stable/hdfs_design.htm
[4] T. White. *"Hadoop: The Definitive Guide."* O′Reilly Media, Inc., 2009.
[5] A. Vulimiri, C. Curino, P. B.Godfrey, T. Jungblut, J. Padhye and G. Varghese, "Global Analytics in the Face of Bandwidth and Regulatory Constraints," in *12th USENIX Symposium on Networked Systems Design and Implementation. NSDI'15*, p. 323–336, Oakland, CA, 2015.
[6] Z. Guo, G. Fox and M. Zhou, "Investigation of data locality in MapReduce," in *12th International Symposium on Cluster, Cloud and Grid Computing. CCGrid'12*, p.419–426,Ottawa, Canada, May 2012.
[7] C. Y.Wang, T. L.Tai, J. S.Shu, J. B.Chang and C. K.Shieh, "Federated MapReduce to Transparently Run Applications on Multicluster Environment," in *BigData Congress*, p. 296–303, 2014.
[8] P. Kondikoppas, C. H.Chiu and S. J.Park,, "MapReduce Performance in Federated Cloud Computing Environment," *Springer High Performance Cloud Auditing and Applications.*, pp. 301-322, August 2013.
[9] Y. Luo, B. Plale, J. Qiu and W. W.Li, "A Hierarchical Framework for Cross-Domain MapReduce Execution," in *Second International Workshop on Emerging Computational Methods for the Life Sciences. ECMLS'11*, p. 15-22, New York, NY, USA, 2011.
[10] L. Wang, L. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen and D. Chen, "G-Hadoop: MapReduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, vol. 29,March 2013.
[11] Q. Zhang, L. Liu, A. Singh, N. Mandagere, S. Gopisetty, G. Alatorre, K. Lee and Y. Zhou, "Improving Hadoop Service Provisioning in a Geographically Distributed Cloud," in *7th International Conference on Cloud Computing. Cloud'14*, p. 432-439, Anchorage, AK, USA, 2014.
[12] S. Imai, S. Patterson and C. A.Varela, "Cost-Efficient High-Performance Internet-Scale Data Analytics over Multi-cloud Environments," in *International Symposium on Cluster, Cloud and Grid Computing. CCGrid'15*, p. 793-796, Shenzhen, China,2015.
[13] S. Kailasam, P. Dhawalia, S. Balaji, G. Iyer and J. Dharanipragada, "Extending MapReduce across Clouds with BStream," *IEEE Transactions on Cloud Computing*, vol. 2, pp. 362–376, July,2014.
[14] P. A.R.S.Costa, X. Bai, F. M.V.Ramos and M. Correia, "Medusa: An Efficient Cloud Fault-Tolerant MapReduce," in *16th International Symposium on Cluster, Cloud and Grid Computing CCGrid'16*, p. 443-452,May 2016.
[15] F. Jrad, J. Tao, I. Brandic and A. Streit, "Multi-dimensional Resource Allocation for Data-intensive Large-scale Cloud Applications," in *4th International Conference on Cloud Computing and Services Science. CLOSER'14*, p. 691-702, Barcelona, Spain, April 2014.
[16] A. Iordache, C. Morin, N. Parlavantzas and P. Riteau, "Resilin: Elastic MapReduce over Multiple Clouds," in *13th International Symposium on Cluster, Cloud, and Grid Computing. CCGrid'13*, p. 261-268, Delft, Netherlands, May 2013.
[17] Z. Hu, B. Li and J. Luo, "Flutter: Scheduling tasks closer to data across geo-distributed datacenters," in *35th International Conference on Computer Communications.INFOCOM'16*, p. 1-9, San Francisco, CA, USA, April 2016.
[18] A. Celesti, F. Tusa, M. Villari and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation," in *3rd International Conference on Cloud Computing.CLOUD'10*, p. 337-345, Miami, FL, USA, July 2010.
[19] E. Hwang and K. H.Kim, "Minimizing Cost of Virtual Machines for Deadline-Constrained MapReduce Applications in the Cloud," in *13th International Conference on Grid Computing.GRID'12*, p. 130-138, Beijing, China, September 2012.
[20] S. Ahn and S. Park, "An Analytical Approach to Evaluation of SSD Effects under MapReduce Workloads," *Semiconductor Technology and Science.*, vol. 15, pp. 511–518, October 2015.
[21] L.A. Wolsey and D.L. Nemhauser. *"Integer and Combinatorial Optimization."* New York, John Wiley 1999.
[22] ILOG CPLEX. *"ILOG CPLEX 11.1 user′s guide."*
[23] R. Fourer, D. Gay, and B. Kernighan. *"Ampl."* Boyd, Fraser, Inc., 1993.
[24] D. Andersen. Theoretical approaches to node assignment. Unpublished Manuscrip. [Online]. Available: http://www.cs.cmu.edu/