Offline Scheduling of Map and Reduce Tasks on Hadoop Systems

Aymen Jlassi[1,2], Patrick Martineau[2] and Vincent Tkindt[2]
[1]*Cyres Group, 19 rue Edouard Vaillant, 37000 Tours, France*
[2]*University François-Rabelais of Tours, CNRS, LI EA 6300, OC ERL CNRS 6305, Tours, France*

Keywords: Big Data, MapReduce Model, Hadoop Scheduling Problem, Time Indexed Formulation.

Abstract: MapReduce is a model to manage quantities massive of data. It is based on the distributed and parallel execution of tasks over the cluster of machines. Hadoop is an implementation of MapReduce model, it is used to offer BigData services on the cloud. In this paper, we expose the scheduling problem on Hadoop systems. We focus on the offline-scheduling, expose the problem in a mathematic model and use the time-indexed formulation. We aim consider the maximum of constraints of the MapReduce environment. Solutions for the presented model would be a reference for the on-line Schedules in the case of low and medium instances. Our work is useful in term of the problem definition: constraints are based on observations and take into account resources consumption, data locality, heterogeneous machines and workflow management; this paper defines boundaries references to evaluate the online model.

1 INTRODUCTION

Manage and access efficiently massive data is becoming more and more important for companies.
Google (Dean, 2004) introduced the model MapReduce as a distributed and parallel Model for data intensive computing. Every job is composed of a set of “map” and “reduce” tasks, which is executed in a distributed fashion over a cluster of machines. Map tasks have to be executed before reduce tasks. Tasks have to be executed as near as possible to the needed data input. Data output of tasks map are transferred to the reduce tasks using the network. MapReduce model is characterized by its simplicity: users wanting to access to data, create “map” and “reduce” tasks, which are next scheduled by specified middleware. The general idea is to schedule those tasks over nodes, which contain data because moving computation near data is less expensive than moving data where computation units are running. For example, in figure 1, average of input set of integers is calculated.
Hadoop (Hadoop, 2005) is one of the most well-known implementation of MapReduce model. It is based on two main components: Hadoop mapReduce and Hadoop distributed file system. The computation level (mapReduce) is composed of three elements. It assures synchronization over different elements and distributes resources between jobs. The Node Manager (NM) is the responsible for resources exploitation per slave machine. The Application Master (AM) is responsible for managing the lifecycle of a job; it negotiates with the RM to obtain needed resources (containers) and manages the execution of job’s tasks.
Hadoop distributed file system (HDFS) is composed of NameNode (NN) as a server and DataNode (DN) as a slave. Files in HDFS are from megabytes up to terabytes size. The number of map tasks depends on the number of chunks of data (Zhou, 2012), one map per data block slice. When the scheduler cannot assign tasks to machines where data are stored, bandwidth on the network is allocated to migrate blocks towards. This paper presents an offline model of scheduling problem on Hadoop with mathematical programming based on the time-indexed formulations which received much attention due to its important impact on approximation algorithms and the quality of its linear programming relaxation.

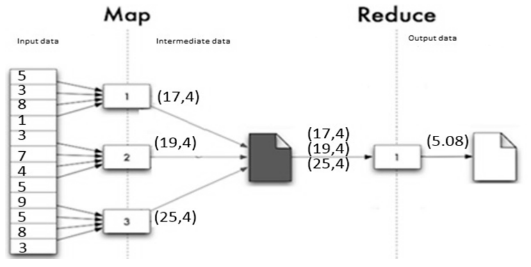

Figure 1: Example of mapreduce job's execution.

It is often used in optimization and approximation for machine scheduling problems. Besides, its linear relaxation yields concise lower bounds than bounds obtained by other integer programming formulations (Queyranne, 1997). Work like (Sousa, 1992) and (Lionel, 2013) argue that scheduling algorithms using LP-relaxation of time-indexed formulations have a constant ratio on their worst-case performance in parallel machine scheduling problems. Researches on the online version of problem suffer from a lack of evaluation: how the efficiency of online algorithms can be evaluated? One way to answer this question is to consider the offline version of the problem, its' optimal solution can be considered as "ideal" reference schedules for online algorithms. In this work, the main motivation is to compute optimal solutions for medium instances of the offline problem.
The remainder is introduced as follows. Section II presents the offline problem of scheduling "map" and "reduce" tasks. In Section III its' mathematical model is introduced. Data generation and model evaluation is presented in Section IV. Section V summarizes the related work. Lastly, Section VI concludes the paper and provides directions of future work.

2 RELATED WORK

The scheduling problem in Hadoop is widely treated in the literature: (Lim, 2014) present a constraint programming formulation of the problem. The objective of the model is to minimize the number of late jobs, which is characterized by its service level agreements (SLA). Authors consider the scheduling of mapReduce jobs comprising an earliest start time, execution time and end-to-end deadline. In this work, authors take into consideration only compute resources (slots), neither RAM nor hard disk are considered. They neglect the relation between data and tasks locations that present a foundation for the map reduce programming model. The work in (Verma, 2012) implements a deadline-based scheduler; it is based on a general model for computing performances bounds on makespan of a given set of n tasks that are processed by k servers (slots). The assignment of tasks to slots is done using an online greedy algorithm; it assigns each task to the slot, which has finished its running task the earliest. (Evripidis, 2014) and (Lin, 2013) propose models, which aim to minimize the total weighted completion time. The first considers that each job has at least one map and one reduce task and each job has at most one task pre-assigned to each processor.

Table 1: Used Notations in the Hadoop scheduling problem.

General data:	
M	The number of machines
N	The number of tasks
N^m	Number of map tasks
N^r	Number of reduce tasks
L^m	Set of map tasks
L^r	Set of reduce tasks
A^b	Set of blocks on the cluster
T	The scheduling horizon
For machines	
m_j^s	The number of slots on machine j $(= m_j^{Sr} + m_j^{Sm})$
m_j^{Sr}	The number of reduce slots on machine j
m_j^{Sm}	The number of map slots on machine j
m_j^r	The quantity of RAM of machine j
m_j^h	The hard drive capacity of machine j
$v_{s,j}$	The CPU frequency associated to the slot s of machine j
v_j	The CPU frequency of machine j $\left(v_j = \sum_{k=1}^{m_j^s} v_{k,j}\right)$
α_j^r	The cost of the use of one unit of ram (1 Mb) per machine j
α_j^h	The cost of the use of one unit of hard drive capacity (1 Mb) per machine j
$\alpha_{s,j}^c$	The cost of the use of CPU on slot s of machine j
For tasks (map, reduce, Application node)	
n_i^r	The quantity of RAM required by task i
n_i^h	The quantity of hard drive required by task i
n_i^b	The number of data block's manipulated by task i
B_i	List of block numbers manipulated by task i
$b_{i,i'}$	Maximum bandwidth between tasks i and i'
n_i^p	Number of tasks preceding task i
E_i	Set of task numbers that must be completed before task i start.
$p_{i,s}^j$	Estimated processing time of task i if processed on slot s of machine j
For HDFS	
S	The size of a data block in the cluster.
r_b	Number of replication block b.
D_b	Set of machines on which block b is located.
bwd	Bandwidth allocated for migrating a block through the network
For the Network	
$G=(V,E)$	The graph modeling the network
b_{max}	The maximum bandwidth associated to any edge $e_u \in E$
P	A set of paths between machines, a path being a set of edges e_u
P_u	The set of couples of machines (j, j') which use the edge e_u

The second considers task pre-assignment to machines and each machine can execute one task at a time. It models the data transfer from map to reduce tasks and it considers map and reduce dependency. (Kodialam, 2012) express the scheduling problem as an optimization problem using linear programming, they aim to minimize the total weight completion time of jobs, they base their work on a set of assumption: machines can process at most one task at time, when a set of tasks is assigned to a processor at the same moment; tasks can be preempt. Fotakis et al. (Fotakis, 2014) consider the case of unrelated processors with multiple Map and Reduce tasks per job. They consider that tasks can be preempted. They present the first polynomial time approximation algorithm, it minimizes the total weighted completion time. However they neglect the data management aspect and they don't consider multiple tasks execution per machine. In this work we associate resources constraints, network bandwidth management to the data flow management.

3 THE OFFLINE SCHEDULING PROBLEM

We summarize in Table 1 the data used in the scheduling model. It is based on four principal parts: the first describes the information about machines and the cost of every resource's use. The second part describes tasks consumption. The third part gives information about data blocks and the fourth describes networks architecture. We consider non-pre-emptible tasks because, in practice, tasks will not be interrupted in Hadoop and when a task fails, it will rerun as it is newly submitted.
Notice: we assume that bandwidth is booked on the network from the end of map tasks until the end of the reduce tasks. The bandwidth reservation avoids delaying job execution when reduce tasks need to communicate with maps machines to ensure some needs (system files, recovers broken data chunks) (White, 2012).

4 A MATHEMATICAL FORMULATION

This section presents a time-indexed formulation of offline scheduling problem in Hadoop. Let us review the formal definition of the model. We adapt the interval-relaxation method proposed in (Dyer, 1990) in single machine case, and in (Schulz, 2002) in multiple machines, with the context of MapReduce model. The time horizon T is divided into a set of irregular intervals. These intervals are defined by the potential dates of starting and finishing execution of tasks. For example, in Figure 2, for $\delta \in [\![0, n-1]\!]$, the intervals $(t_\delta, t_{\delta+1}]$ are used to execute tasks, where $t_\delta \in [0, T]$.

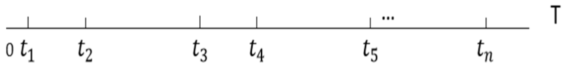

Figure 2: Presentation of the index over time.

We use the following variables:

$$x^{j}_{i,s,t_\delta}\begin{cases} q, & \text{the amount of time period, the task i is processed on slot s of the machine j in } (t_\delta, t_{\delta+1}] \\ 0, & \text{otherwise} \end{cases}$$

Thus $x^{j}_{i,s,t_\delta}/p^{j}_{i,s}$ specify that the task is being processed on machine j during the time interval $(t_\delta, t_{\delta+1}]$.

$$y^{j,j'}_{b,t_\delta}\begin{cases} 1, \text{if block b is on machine j at } [t_\delta, t_{\delta+1}[\text{ after a migration from } j' \\ 0, \text{otherwise} \end{cases}$$

$$u^{j,j'}_{b,t_\delta}\begin{cases} 1, \text{if block b is being migrated from machine j to } j' \text{ at } [t_\delta, t_{\delta+1}[\\ 0, \text{otherwise} \end{cases}$$

$$z^{j,j'}_{l,l',t_\delta}\begin{cases} 1, \text{if a map task } l' \text{ is processed on machine } j' \text{ and is finished at time t and a reduce task l is processed on machine j and finished after } t_\delta. \\ 0, \text{otherwise} \end{cases}$$

We refer to TST as the total time spent for processing all tasks on the cluster and TRC as the total resource cost induced by the execution. The scheduling problem in Hadoop can be modeled with the objective functions (1) and (2). The TST (1) considers the total execution time of tasks (the first term on the left-hand side of the equation) and the time of data transfer between map and reduce tasks (the second term on the right-hand side of the equation). The TRC (2) considers the resources machines' cost when processing tasks (the first term on the left-hand side of the equation) and the use of resources due to data transfer (the second term on the right-hand side of the equation). The constraints of the model are classified in three categories: resource constraints, processing constraints and the network constraints.

$$\text{Minimize TST} = \sum_{t_\delta=1}^{T}\left[\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{s=1}^{m_j^s}\left(x^j_{i,s,t_\delta}\big/p^j_{i,s}\right) + \sum_{i\in L^r, l\in E_i}\ \sum_{j,j'=1}^{M}\ \sum_{b\in A^b} y^{j,j'}_{b,t_\delta}\left(S\big/b_{l,i}\right)\right] \quad (1)$$

$$\text{Minimize TRC} = \sum_{t_\delta=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{s=1}^{m_j^s} x^j_{i,s,t_\delta}\left[\alpha_j^r n_i^r + \alpha_j^h n_i^h + \alpha^c_{s,j}\right] + \sum_{t_\delta=1}^{T}\sum_{j=1}^{M}\sum_{j'=1}^{M}\sum_{i=1}^{N}\sum_{b\in B_i}\left(y^{j,j'}_{b,t_\delta} + u^{j,j'}_{b,t_\delta}\right)\left(\alpha_j^r n_i^r + \alpha_j^h n_i^h\right) \quad (2)$$

In the subsection 4.1, constraint (3) guarantees that no more memory than available is used. Constraints (4) and (5) guarantee that the number of reduce (resp. map) tasks running on machine j at time t is less than the number of reduce slots (resp. map slots). Constraint (6) ensures that the overall local disk space used (by the assigned tasks and migrated data) cannot exceed the availability of each machine. In the subsection 4.2, the inequality (7) guarantees the precedence relation between map and reduce tasks associated to the same job are satisfied. If we have many map tasks, reduce tasks are scheduled after the schedule and the end of all map tasks. In figure 1, we compute average of input data, we will have wrong result if reduce tasks start before the end of map tasks. Constraints (7) and (8) ensure that all map tasks (resp. reduce tasks) must be processed.

In the subsection 4.1, the constraints define the policy of data blocks management in Hadoop. The inequality (10) specifies if block b is stored in HDFS on machine j. The constraints (11) and (13) impose the relation between y's and u's variables, constraint (13) triggers data migration to ensure that block must be available on the machine before a map task starts and constraint (11) ensures if it is available on a machine after it has been migrated. The Inequalities (12) disable the start of map tasks (imposed by the constraint 8) if the manipulated blocks are not present on the machine on which they have been assigned. The inequalities (15) enable to fix the values of the $z^{j,j'}_{l,l',t_\delta}$ variables. When the tasks map and reduce are on the same machine, we don't have network communication and the right part of inequality (15) will be 0.

4.1 Resources Constraints

$$\sum_{s=1}^{m_j^s}\sum_{i=1}^{N} n_i^r\left(x^j_{i,s,t_\delta}\big/p^j_{i,s}\right) \le m_j^r \quad (3)$$
$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T$$

$$\sum_{i\in L^r}\left(x^j_{i,s,t_\delta}\big/p^j_{i,s}\right) \le 1 \quad (4)$$
$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T, \forall s = 1 \dots m_j^{Sr}$$

$$\sum_{i\in L^m}\left(x^j_{i,m_j^{Sr}+s,t_\delta}\big/p^j_{i,m_j^{Sr}+s}\right) \le 1 \quad (5)$$
$$\forall j = 1 \dots M, \forall t_\delta = 1 \dots T, \forall s = 1 \dots m_j^{Sr}$$

$$\sum_{s=1}^{m_j^s}\sum_{i=1}^{N}\left(n_i^h x^j_{i,s,t_\delta}\big/p^j_{i,s}\right) + \sum_{j'=1,j'\neq j}^{M}\ \sum_{i=1}^{N}\sum_{b\in B_i} S(b)\left(y^{j,j'}_{b,t_\delta} + u^{j,j'}_{b,t_\delta}\right) \le m_j^h \quad (6)$$
$$\forall\, j = 1 \dots M;\ \forall t_\delta = 1 \dots T$$

4.2 Tasks Constraints

$$n_k^p * x^j_{i,s,t_\delta}\big/p^j_{i,s} \le \sum_{u=1}^{M}\sum_{s'=1}^{m_u^{Sm}}\sum_{t'=0}^{t_\delta}\sum_{l\in E_k}\left(x^u_{l,m_u^{Sr}+s',t'}\big/p^u_{l,m_u^{Sr}+s'}\right) \quad (7)$$
$$\forall k \in L^r,\ \forall t_\delta = 0 \dots T-1, \forall\, j = 1 \dots M, \forall\, s = 1 \dots m_j^{Sr}$$

$$\sum_{j=1}^{M}\sum_{s=1}^{m_j^{Sm}}\sum_{t_\delta=0}^{T-1}\left(x^j_{i,m_j^{Sr}+s,t_\delta}\big/p^j_{i,m_j^{Sr}+s}\right) = 1 \quad (8)$$
$$\forall\, l \in L^m$$

$$\sum_{j=1}^{M}\sum_{s=1}^{m_j^{Sr}}\sum_{t_\delta=0}^{T-1}\left(x^j_{i,s,t_\delta}\big/p^j_{i,s}\right) = 1 \quad (9)$$
$$\forall\, l \in L^r$$

4.3 Constraints Associated to the Migration of Data Blocks

$$y_{b,t_\delta}^{j,j} = \begin{cases} 1, \forall j \in D_b \\ 0, \forall j \notin D_b \end{cases} \quad \forall t_\delta = 0 \ldots T, \forall b \in A^b \tag{10}$$

$$u_{b,t_{\delta-1}}^{j,j'} \le y_{b,t_\delta}^{j,j'} \tag{11}$$

$$\forall\, b \in A^b; \forall t_\delta = 1 \ldots T-1; \forall\, j, j' = 1 \ldots M; j \neq j'$$

$$u_{b,t_\delta}^{j,j'} \le \sum_{l \in L^m} \sum_{t'=t_\delta}^{T-1} \sum_{s=1}^{m_j^{Sm}} \left(u_{b,t}^{j,j'} x_{l,m_j^{Sr}+s,t'}^{j} \middle/ p_{l,m_j^{Sr}+s}^{j} \right) \tag{12}$$

$$\forall\, b \in A^b; \forall t_\delta = 0, \ldots, T-1; \forall\, j, j' = 1 \ldots M, j \neq j'$$

$$\sum_{s=1}^{m_j^{Sm}} \left(x_{l,m_j^{Sr}+s,t_\delta}^{j} \middle/ p_{l,m_j^{Sr}+s}^{j} \right) \le \sum_{j'=1}^{M} y_{b,t_\delta}^{j,j'} + u_{b,t_\delta}^{j,j'} \tag{13}$$

$$\forall l \in L^m;\ \forall\, b \in B_l; \forall t_\delta = 0, \ldots, T-1; \forall\, j = 1 \ldots M$$

4.4 Network Constraint

These constraints define the use of the network in terms of bandwidth. Constraint (14) imposes that all consumed bandwidth (for migration and transfer of data) is less than the maximum bandwidthb_{max}.

$$\sum_{(j,j') \in P_e} \left[bwd \sum_b u_{b,t_\delta}^{j,j'} + \sum_{i \in L^r} \sum_{l \in E_i} \sum_{s=1}^{m_j^s} z_{i,l,t_\delta}^{j,j'} * b_{i,l} \right] \le b_{max} \tag{14}$$

$$\forall e \in E;\ \forall t_\delta = 1, \ldots, T-1$$

$$\sum_{s=1}^{m_j^{Sr}} \left(x_{l,s,t''}^{j} \middle/ p_{l,s}^{j} \right) + \sum_{s=1}^{m_{j'}^{Sm}} \left(x_{l',m_j^{Sr}+s,t'}^{j'} \middle/ p_{l',m_j^{Sr}+s}^{j} \right) - 1 \le z_{l,l',t_\delta}^{j,j'} \tag{15}$$

$$\forall l \in L^r, \forall l' \in E_l;\ \forall t_\delta = 0, \ldots, T-2; \forall t' = 0, \ldots, t_{\delta-1}; \forall t'' = t_\delta, \ldots, T-1; \forall\, j, j' = 1 \ldots M, j \neq j'$$

5 EXPERIMENTATION

This article implements a model and tries to find solutions using CPLEX mathematic solver. Face to the multi-criteria property of the problem, the model is concentrated on the time execution aspect and neglects cost execution of the job. It uses an experiment setting for the evaluation of the model using the methodology in (Lionel, 2013). Data input of the model presents an important deal and imitates real world tasks executions. Machine configuration is extracted from AWS (Aws, 2014) and portioned in three categories of machines. Tasks information depends on the size of data input computed by every task. In order to evaluate the persistence of the model, we generate randomly four input data concerning tasks following uniform law: memory, disk consumption, the time execution per task and location of data blocks (Gupta, 2013). We generate also network and cluster configuration details. Table 2 synthetizes values of the expected data input of machines. The first column indicates the category of the machine.

The second column indicates the number of core CPU on the machine. The third one contains the amount of memory per machine. The column number four indicates the quantity of hard disk in Gb. The fifth column contains the frequency of one core CPU on the machine. The sixth column indicates the bandwidth allocated for network communication.

Table 2: Types of generated physical machines.

Category	CPU node	RAM (Gb)	SSD (Go)	CPU freq per core (GHZ)	Bdw (GB)	α_j^r	α_j^h	$\alpha_{s,j}^c$	Slots map	Slots reduce
c3.2xlarge: co pute optimized	8	15	160	2.8 Intel Xeon E5-2680v2	1	1	1	2	5	2
i2.2xlarge: storage optimized	8	61	1600	2.5 Intel Xeon E5-2670v2	1	3	5	2	4	3
r3.xlarge: memory optimized	4	30.5	160	2.5 Intel Xeon E5-2670v2	2	2	1	1	2	1

Table 3: Characteristics of used jobs.

Job	Tasks reduce	Tasks map	Type of Job
1	2	3	--
2	2	6	--
3	3	9	--

Columns number seven, eight and nine indicate respectively the unit cost of the memory use (unit = 16Mb), hard disk (unit = 1Gb), and a core of CPU. Despite the evolution in Hadoop, we adopt the principle of separation between slots; the last two columns contain the number of reduce and map cores (slots) per machine. The costs of resources consumption are expressed in columns seven, eight and nine and they depend on the type of machine. We generate the completion time needed to treat tasks; these values depend on the size of the block. We define:

$$P^{j}_{l,s} = \mathrm{TimeStartUpVM} + S * nt(l) * \left[vs(l,j,s) \big/ 10 * \mathrm{SpeedProcessorRate} \right] \quad (16)$$

We take into account the needed time to start up virtual machines TimeStartUpVM, the size of block and the amount of data computed per GHz per unit of time SpeedProcessorRate. We benefit from the last variable to inject the random aspect depending on the categories of machines: for the category "compute optimized", $\mathrm{SpeedProcessorRate} \in [160,320]$ for the other types $\mathrm{SpeedProcessorRate} \in [80,160]$.The estimation of memory (n^r_i) and hard disk consumption (n^h_i) depends on the type of the job. Table 4 summarizes used formulas in the generation of data related to the three types of jobs: the number of tasks per job is relatively limited; CPLEX limitation imposes this choice of number of task per job face of the use of one big job. We inject random values at many levels of the data input generation. Face to the large quantity of data generated by the model in time indexed formulation, we consider S=64Mb and its replication is equal to one. We consider the same size (S) and replication properties of data blocks however we generate randomly the location of the blocks on machines. The network bandwidth for block migration is fixed by the formula $\mathrm{bwd} = \min[S * 0.2, 128]$. Network is generated as a binary tree. We repeat the following process: at the main node, we generate a switch; its left child node will be one physical machine selected randomly, the right child will be another switch and so on until all physical machines will be placed on the binary tree.

Table 5 describes scenarios used for the model's test. For each scenario, we randomly generate 20 instances. The time horizon depends on scenarios and it is divided in intervals. To find the correct value of time horizon, we define an upper bound for every solution using this formula (17). If there is no solution for a particular value of the time horizon, we increment time horizon by a unit of time. We consider that an interval ($[t_\delta, t_{\delta+1}[$ from figure 2) is sufficient to transfer data block between machines. In conclusion, we limit bandwidth threshold to migrate blocks and we limit the transfer duration of a block to one interval. To compute the real duration's value of a schedule per scenario, we define "RealTime" (formula 18) as the real time needed to execute tasks in a solution.

$$T = \mathrm{integer}\left(\frac{N^m}{\mathrm{TotalSlotMap}} + \frac{N^r}{\mathrm{TotalSlotReduce}}\right) + 2 \quad (17)$$

$$\mathrm{RealTime} = \sum_{t=0}^{T} \max_{\substack{j=1..M \\ s=1..ms(j) \\ l \in L}} x^{j}_{l,s,t} \quad (18)$$

$$\mathrm{RealValueOfTimeHorizonUnit} = \mathrm{RealTime}/T \quad (19)$$

"RealTime" is a posterior computation, after the compute of the scheduling solution.

Table 4: Basic formulas to generate memory and hard disk consumptions per task.

Type of Job	$n^r_i = n^b_i * S * XY$	$n^h_i = (n^b_i * S * WZ)/1024$
(1) CPU intensive	$XY \in [0.3,0.6]$	$WZ \in [13,26]$
(2) RAM intensive	$XY \in [0.4,0.8]$	$WZ \in [30,46]$
(3) I/O intensive	$XY \in [0.6,1]$	$WZ \in [46,76]$

Table 5: Different scenarios for the generation of tasks, machines and blocks input data.

Scenarios	N1	N2	N3	M1	M2	M3	Blocks	N	M	T
1	1	1	0	1	1	1	10	13	3	3
2	3	0	0	0	2	0	10	15	2	3
3	1	1	1	1	1	1	10	25	3	9
4	3	3	0	0	0	2	10	39	2	15
5	6	0	0	1	1	1	10	30	3	7
6	2	3	1	1	1	0	10	46	2	15
7	3	1	1	0	2	0	10	35	2	6

Table 6: Computational results (20 instances per scenario).

	#InFeas	#Solved	MemLimit	TimLimit	N_{min}	N_{avg}	N_{max}	T_{min}	T_{avg}	T_{max}	Real value of unit of T
Sc1	0	20	0	0	0	7.95	132	0	0.45	1	85.66
Sc2	0	20	0	0	27	42.5	164	0	19.04	174	95.66
Sc3	1	18	0	1	4	5357.6	62168	6	97.75	1044	122.4
Sc4	3	16	1	0	0	6675.15	28365	16	292.36	1313	54,8
Sc5	2	18	0	0	40	132.8	1791	10	66.9	757	94.62
Sc6	4	15	0	1	115	142	389	28	185	1641	126.23
Sc7	0	20	0	0	3	61.25	193	5	10.1	22	70.53

It is used to compute the real duration to execute jobs in a scenario. We define established value as the time Horizon T per scenario; we compute a value of a unit of T as regular time horizon with the formula (19). We enumerate the minimum, maximum and average of the RealTime over iterations and we choose the maximum value to compute the value of a unit of T per scenario. This value is used in the evaluation of the results of solutions.
To test the model, we use a PC with an Intel(R) Core (TM) i5-3360M CPU with 4 cores at 2.8 GHz and 4 Gb of RAM. The linear program formulation has been solved by CPLEX 12.2 with parallel solve (4 threads) and limit time 1800 seconds and memory limit of 2 Gb of RAM. When the time limit or the memory limit is reached, the given solution of the instance will be declared unsolved. Otherwise, CPLEX will return the best solution. For each scenario, table 6 presents: the number of infeasible instances (column #InFeas), the number of instances solved to optimality (column #Solved). The number of instances on which CPLEX stops due to the memory limit (column Mem) and the number of instances on which CPLEX stops due to the time limit (column Time). The columns from number six to number eight provide the minimum, maximum and average number of nodes explored by CPLEX in its branch and cut algorithm while solving the problem. There is no relation between the number of machines and the number of explored nodes. Scenarios 4 and 6 have two machines each, however the number of explored nodes in scenario 4 is largely higher than the number of nodes explored in scenario 6. In the same topic, the number of explored nodes is independent from the number of scenario 7 for example has a number of tasks to schedule higher than scenario 5. However, the number of node explored in scenario 5 is higher than in scenario 7. The columns from number nine to number eleven provide minimum, average and maximum CPU time (in seconds) taken by CPLEX to solve instances. In this topic, we consider only instances, which have infeasible or feasible results. The result shows that there are large disparities concerning CPU times used to find solution. The last column presents the real value of the time horizon unit; it is used as a comparison reference. It is extracted from the approximate value of the average completion time per scenario. Results of founded schedule time of a scenario argue that it depends on the number of tasks and machines; Scenarios 4 and 6 have largest value of the time horizon. These scenarios have the largest number of tasks to schedule. Scenarios 1 and 2 have the smallest number of tasks and the smallest number of machines in an instance. Results are function of the number of tasks and the number of machines in an instance and some instances take more time to find solution than others. Scenario 6 for example schedules 46 tasks on two machines; it has the largest value of completion time.

6 CONCLUSIONS

In this paper, we propose an offline mathematical model for the scheduling problem in Hadoop. Two kinds of tasks are considered: “map” and “reduce” tasks with dependencies between them. This paper also presents an in-depth study of the major aspects of MapReduce model, such as tasks dependency, network consumption, data flow management and the non-interruptive tasks executions.
It aims at scheduling tasks with the minimum cost of used resources and the minimum total processing duration. We merely focus on a pure scheduling problem; we propose an offline model assuming that all data are known. We present a realistic model, which considers dependence between tasks. We consider data locality and we model data migration and transfer between heterogonous machines. All considered constraints emulate the real world environment in Hadoop. Heterogeneous machines cluster and possibility to execute many tasks per machine are also considered. The proposed model is based on a time-indexed formulation, which despite

its pseudo polynomial number of variables. It has already been shown as an efficient formulation compared to other integer programming formulations. We use the commercial solver CPLEX to find the optimal solution for small and medium size of instances. We give community a boundary to reference with and to evaluate their scheduling algorithms for this size of instances. It turns out that the offline problem is interesting in it self and can be used to design good online strategies. Solution for this model would be a reference for the on-line schedules in smaller dimension to validate first result. Future work will deal with the online aspect concerning the scheduling problem; we plan to propose a heuristic solution and use this work in the evaluation.
Online solution considers at first Total completion time, in a second time we take into account the resources consumption (energy) in a multi-criteria scheduling aspect.
The final solution will be implemented over Hadoop simulation system and evaluated in a large scalability face to default scheduler in Hadoop.

ACKNOWLEDGMENTS

This work was sponsored in part by the CYRES GROUP in France and French National Research Agency under the grant CIFRE n°2012/1403.

REFERENCES

Aws. 2014. Instances-types. Retrieved from Aws: http://aws.amazon.com/fr/ec2/instance-types/
Dean, J., & Ghemawat, S., 2004. MapReduce: Simplified Data Processing on Large Clusters. In Communications of the ACM.
Dyer, M. E., & Wolsey, L. A., 1990. Formulating the single machine sequencing problem with release dates as a mixed integer program.
Evripidis Bampis, V. C., 2014. Energy Efficient Scheduling of MapReduce Jobs. In 20th International Conference.
Fotakis, D., Milis, I., & Zampetakis, E., 2014. Scheduling MapReduce Jobs on Unrelated Processors. In the Workshop Proceedings of the EDBT/ICDT.
Gupta, S., Fritz, C., Price, R., Hoover, R., de Kleer, J., & Witteveen, C., 2013. Throughput Scheduler: learning to schedule on heterogeneous Hadoop clusters. In (ICAC '13), International Conference on Autonomic Computing.
Hadoop Project, 2005. (A. foundation, Producer) Retrieved from http://hadoop.apache.org/
Kodialam, M. S., Lakshman, T., Mukherjee, S., Chanwg, H., & Lee, M. J., 2012. Scheduling in mapreduce like systems for fast completion time. In Patent Application Publication.
Lim, N., Majumdar, S., & Ashwood-Smith, P., 2014. A Constraint Programming-Based Resource Management Technique for Processing MapReduce Jobs with SLAs on Clouds.
Lin, M., Zhang, L., Wierman, A., & Tan, J., 2013. Joint Scheduling of Processing and Shuffle Phases in MapReduce Systems. In P. o. Conference (Ed.).
Lionel, E.-D., Adrien, L., Patrick, M., Ameur, S., Vincent, T., & Denis, T., 2013. A Server Consolidation Problem: Definition and Model. In Proceedings of the 14th conference ROADEF.
Queyranne, M., & Schulz, A., 1997. Polyhedral Approaches to Machine Scheduling. In Mathematical Programming.
Schulz, A. S., & Skutella, M., 2002. Scheduling Unrelated Machines by Randomized Rounding. In SIAM Journal on Discrete Mathematics.
Sousa, J. P., & Wolsey, L. A., 1992. A time indexed formulation of non-preemptive single machine scheduling problems. In Mathematical Programming.
Verma, A., Cherkasova, L., Kumar, V. S., & Campbell, R. H., 2012. Deadline-based Workload Management for MapReduce Environments: Pieces of the Performance Puzzle.
White, T., 2012. Hadoop, The Definitive Guide (3rd Edition ed.). O'REILLY. 3rd edition.
Zhou, W., Han, J., Zhang, Z., & Dai, J., 2012. Dynamic Random Access for Hadoop Distributed File System. In (ICDCSW), Distributed Computing Systems Workshops.