Static Multi-tier MapReduce Jobs Workflow Scheduling in Hybrid Clouds

Hani Al-Sayeh TU Ilmenau, Germany
hani-bassam.al-sayeh@tu-ilmenau.de

ABSTRACT

Nowadays, huge data is being produced and needs to be analyzed to increase economic profits, improve civil services and achieve furthermore objectives. Processing Big Data requires multiple analysis stages represented as workflows of big data jobs. In spite of the great evolution of big data processing techniques, the lack of computing resources is one of the most significant challenges, that's why utilizing and extending on demand these resources, using hybrid clouds, is highly required. Good workflow scheduling approaches leads to better utilization of resources, reducing costs and meeting deadline constraints. To achieve this goal, we are proposing an architecture of multi-tier MapReduce jobs workflow scheduling in hybrid clouds with a basic cost model.

Keywords

MapReduce, Workflow scheduling, Hybrid clouds

1. INTRODUCTION

During the last years, the amount of produced data is increasing dramatically. Due to the great evolution of sensing technologies and E-commerce systems, data production is accelerating more and more especially in astronomy, telecommunication, social media and many other fields. Nowadays, the need for processing this huge amount of data is becoming higher day after day to increase economic profits, improve civil services and achieve furthermore objectives.

Traditional data analysis systems and infrastructures are not sufficient to handle the Big Data processing new requirements like high throughput, low latency, meeting deadlines and others. That's why many concepts have been presented to process data in multiple machines concurrently e.g. data partitioning and replication, query parallelization, and resource utilization. Many frameworks and paradigms are introduced to implement these concepts, Google has proposed the most famous one named MapReduce.

Many of recent scientific applications require multiple processing stages to handle huge streams of data, these stages can be represented as a workflow of tasks. To facilitate scientific workflows deployment and hide infrastructure complexity during executing them, cloud computing is presented as a cost-efficient computing infrastructure. There are several types of clouds [1]:

- Private clouds: Limited computational resources that users own and use to run their applications.
- Public clouds: Huge collection of computational resources that users rent to run their applications. Theoretically, these resources can be considered as unlimited ones.
- Hybrid clouds: A combination of both previous cloud types.

Relying on MapReduce processing model and cloud computing techniques, analyzing and extracting valuable information from big data sources within a feasible time is possible. But, there are still many challenges that must be taken into account during designing Big Data processing solutions. the most critical ones are meeting deadlines and the lack of computational and storage resources. To come over these challenges, good tasks scheduling in a hybrid cloud is required. We will discuss our proposed multi-tier scientific workflow scheduling on hybrid clouds taking into consideration that each task in the workflow can be a MapReduce job. The rest of the paper is organized in the following way. Section 2 discusses main background concepts. Related work and Challenges are discussed in Section 3 and 4, respectively. The multi-tier scheduling model and architecture are proposed in section 5. A conclusion is written in sections 6.

2. BACKGROUND

In this section, we will discuss main concepts required for scheduling scientific workflows in clouds.

2.1 Cloud computing

Cloud computing is large scalable computational and storage resources that provide data processing services over the Internet [1]. Clouds provide unlimited resources which are relatively cheap, and let users to pay for the only used resources instead of reserving them more than needed. Also, clouds use the visualization concept, which allows users to run their application without considering the hosting OS requirements and configuration efforts. In addition, clouds hide technical details like network maintenance, data backups, failure recovery and others. Infrastructure as a Service

30th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 22.05.2018 - 25.05.2018, Wuppertal, Germany.

(Iaas) is one of cloud services that allows users to rent computation and storage resources and run their applications. IaaS is a suitable and cost-efficient solution for on demand extending limited resource in private clouds.

2.2 Scientific workflows

Scientific applications require multiple data processing stages which might be executed sequentially, or in parallel when there is no data dependency in between. Dividing the main task into sub-tasks is required for efficient tasks scheduling in multiprocessors clusters. Data dependencies between subtasks can be represented using Direct Acyclic Graph DAG [5]. Fig.1 Shows an example of a Directed Acyclic Graph DAG.

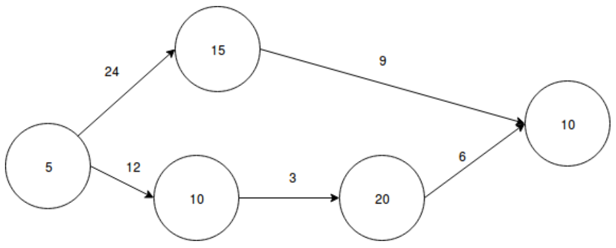


Figure 1: Directed Acyclic Graph (DAG).

2.3 Big Data Processing

Big Data processing and analysis requires a processing model (as MapReduce) and a storage platform (as HDFS).

2.3.1 MapReduce

MapReduce is a large-scale parallel data processing model on clusters. Data is fragmented into data sets and distributed among the processing nodes where processing operations are done in parallel. It contains the following main stages [2]:

1. Map: Each node processes the locally stored data set using the user-defined map function and store intermediate key-value results on its permanent local storage temporarily.

2. Shuffle: Mappers send each group of key-value pairs to its corresponding reducer. At the end, each reducer has a list of values for each key. Shuffling process relies on the performance of the cluster network.

3. Reduce: Processing nodes receive a list of values for each key and run the user-defined reduce function. The reducers output will be accumulated as a final result and written in a pre-defined output file.

2.3.2 HDFS

HDFS is proposed by Google and a stand for Hadoop Distributed File System. It stores huge files by fragmenting them into blocks which are replicated and stored in multiple machines. HDFS contains a single NameNode and multiple DataNodes. Each DataNode stores data blocks and runs MapReduce operations. The NameNode stores data allocation catalog and is responsible for handling users operations on files such as create, delete and rename.

3. RELATED WORK

Many efforts has been devoted to schedule workflows of tasks in different types of clouds. Each cloud type has its own challenges and the proposed state of the art workflow schedulers focuses on one or multiple objectives to come over these challenges. Also, some workflow scheduler systems are provided to manage MapReduce jobs.

3.1 Executing Workflow of MapReduce Jobs

Oozie is a well-known MapReduce workflow design and management application. With its web interface, users can design complex scientific workflows and run them on clusters. In addition, users can design workflows, using other workflow design tools, then Oozie parses the workflows definition file (an XML file) and runs it.

In addition to Oozie, Azkaban (developed by Linkedin) and Luigi are workflow management systems for MapReduce jobs.

3.2 Workflow Scheduling in Clouds

The main objective of the workflow schedulers is to allocate suitable resources for all workflow tasks considering their data dependencies. The great growing of data and the importance of analyzing it motivate a lot of researchers to present workflow schedulers on clouds with respect to different objectives.

3.2.1 Workflow Scheduling in Private Clouds

In private clouds, the resources are limited and utilizing them is the main focus of most research studies to meet single or multiple objectives such as:

- Increasing system throughput: Increasing the number of accomplished workflows in a time period.

- Reducing system latency: Minimizing the makespan of the workflow. Workflow makespan is the time when the execution of its last node finishes.

- Meeting deadlines: Each workflow may have a deadline. The main task of the scheduler is to meet all the deadlines of the submitted workflows. The challenging part is that with limited resources in the private clouds, meeting all deadlines may not be achieved even with the best utilization of resources. That's why an extension of resources might be required.

3.2.2 Workflow Scheduling in Public Clouds

In public clouds, the resources are considered unlimited. But, new challenges shall be taken into account like:

- Data confidentiality: When the analyzed data is confidential, then it is not possible to upload it to public clouds. That's why running applications on public cloud resources is not sufficient when there are security aspects.

- Monetary costs: The user must pay for all the resources that are included in processing workflows. That's why running applications on public clouds is the most expensive compared with private and hybrid ones.

- The speed and robustness of the internet connection: Although the existence of unlimited computing resources speeds up workflows processing operation and minimizes their makespans, additional time required to upload data to public clouds and downloading its results. If the internet connection is slow, then public clouds are not suitable to run big data applications.

Minimizing monetary costs and meeting deadlines are the main objectives for workflow schedulers in public clouds.

3.2.3 *Workflow Scheduling in Hybrid Clouds*

To come over private and public clouds challenges, hybrid clouds solutions are widely used. Workflow scheduling in hybrid clouds plays a balancing role between deadlines and costs constraints. Hybrid cloud optimized cost (HCOC) algorithm [3] is one of famous workflows scheduling approaches that focuses on meeting deadlines and minimizing monetary costs.

3.3 Discussion

Although great effort has been devoted to study workflow scheduling in clouds for several years, most of the studies don't consider scheduling workflows that consist of multiple MapReduce jobs. In addition, although scientific workflows are defined by domain experts, many enhancements and operations can be performed through them to utilize network and computing resources. Additionally, most of studies assume that processors are idle immediately after tasks execution without considering that exchanging data consumes processors' time. Moreover, many of hybrid cloud task scheduling approaches don't study the internet connection between private and public clouds as a critical limited resource which must be shared and utilized wisely Also, these approaches don't take into account the costs for uploading and downloading data volumes.

4. CHALLENGES

Scheduling workflow tasks in clusters is an NP-Complete problem [4] Hence, reaching the global optimal solution is not possible within a feasible time. This issue becomes more complicated during scheduling tasks on hybrid clouds.

In addition, estimating MapReduce jobs costs is still an open issue. It is hard to expect the output volume of each mapper because it depends on the data context. The same job may produce different data volumes even though input files have the same size. Because of that, it is tricky to predict the required budget for running a job on a public cloud. During job execution, it might exceed the budget limits.

Moreover, during the real-time execution, some failures might occur which cause exceeding budget and deadlines.

5. STATIC MULTI-TIER WORKFLOW SCHEDULING IN HYBRID CLOUDS

Our goal is to schedule workflows consisting of multiple MapReduce jobs in hybrid cloud with respect to deadlines and budget constraints. The proposed multi-tier static scheduling approach is a meta-heuristic and an iterative one. In each iteration, it allocates tasks on private and public cloud resources, run them on a simulation tool and evaluate the simulation results. In addition, a cost model for running MapReducer jobs in hybrid cloud is proposed including data uploading and downloading costs and considering the data connection bandwidth as a critical resource shared among public and private clouds processors. Even after finding a suitable allocation plan, additional iterations can be performed to minimize monetary costs and reduce execution time, then with this added cost and time margins the impact of system failures will be reduced.

5.1 Modeling

The DAG model is G=(V,E), where V in the set of tasks (or nodes) and E is the set of edges between them. Each task t_i in V has a number of instructions in_i and a deadline d_i, which is ∞ if the task does not have a deadline constraint. Each edge $e_{ij} \in E$ represents data dependency and volume between task t_i and task t_j, t_j cannot be started until t_i finishes, t_i called parent of t_j, and t_j is a child of t_i. The first node in DAG, which does not have any parent, called the entry tasks. The last node in DAG, which does not have children, called exit task.

Hybrid cloud model is H=(PR, PU, G, UP, DW), where PR is the private cluster resources, PU is the public cluster resources, G=(V,E) is the directed acyclic graph that represents the submitted scientific workflows (combined in one as we will show later), UC is the upload bandwidth and DW is the download one. Private cloud resources are represented as PR=(P,L), where P is the set of processors and L is the set of data transfer links between them. Also, public cloud resources are represented as PU=(P,L). Each link l_{ij} between $p_j, p_j \in P$ has data transfer frequency fr_{ij} (data volume per time unit). Each processor p_i has a computational power cp_i (instructions per second), network interface speed ns_i (data volume per time unit) and monetary costs per time unit mc_i, which is zero for private cloud processors.

5.1.1 *Local Data Transfer Costs*

Transferring data between two processors in the same cluster requires taking data volumes, processors' network interface speeds and the network link capabilities into consideration. Exchanging data consumes processor's time, so it is not enough to take only tasks instruction number into account while calculating processor occupation time. The time that a processor $p_i \in P$ needs to transfer data volume v, whether in sending and receiving, is defined as

$$pdtt_i(v) = \frac{v}{ns_i} \tag{1}$$

To transfer data volume v, each network link $l_{ij} \in L$ between two processors $p_i, p_j \in P$ consumes data transfer time, which is defined as

$$ldtt_{ij}(v) = \frac{v}{fr_{ij}} \tag{2}$$

When processor p_i sends data to processor p_j, both processors take the same amount of time, which is the time taken by the slowest part in data transfer operation. The data transfer time of each processor is defined as:

$$dt_{ij}(v) = \max \left\{ pdtt_i(v), pdtt_j(v), ldtt_{ij}(v) \right\} \tag{3}$$

5.1.2 *Total Task Execution Time*

When a task is allocated on a processor, the processor needs time to receive the required input data, execute the task and send its results. So, the total execution time of a task t_i allocated in processor p_j, is defined as:

$$et_{ij} = dt_{pj}(e_{aj}) + \left(\frac{in_i}{cp_j}\right) + dt_{jl}(e_{jb}) \quad (4)$$

Where t_a, allocated in p_p, is a parent task of t_i, and t_b, allocated in p_l, is a child task of t_i.

5.1.3 Data Transfer Costs Between Clouds

To calculate the time required for uploading and downloading data, the limitation of data exchanging connection bandwidth between public and private clouds which is shared among multiple processors must be taken into account. One of the following two approaches can be selected:

- All processors share the connection bandwidth concurrently. Hence, the bandwidth will be distributed in a round robin fashion among all concurrent data exchanging operations. This approach is recommended when the dta connection speed between public and private clouds is higher than processors' network interfaces speed.
- First coming processor reserve the whole connection bandwidth for transferring data and the others wait in a queue. This approach is recommended when the public-private data connection speed is less than processors' network interface speed.

As highlighted previously in equation (3), The data sending and receiving processors will be occupied for data exchange, and if the data transfer time is slow then both processors will not be ready to process further tasks for a long time. If the internet connection is shared among multiple data exchange operations that waste processors time, that's why the second approach (FIFO) is selected. In this case, even if a processor shall wait for other data exchanging operations, it can store results into its permanent storage medium and process another task until its turn comes to send data. So, the time required for uploading and downloading is defined as:

$$up(v,t) = \frac{DataVolumesInUpQueue(t) + v}{UP} \quad (5)$$

$$dw(v,t) = \frac{DataVolumesInDwQueue(t) + v}{DW} \quad (6)$$

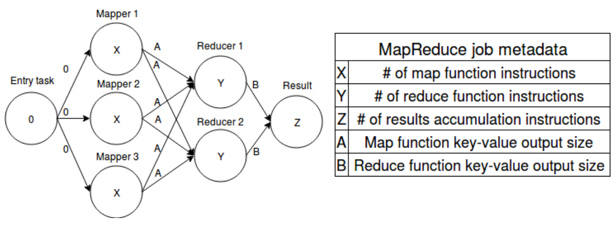
Where v is the volume of exchanged data, t is the timestamp when data is ready to be exchanged, and DataVolumesInUpQueue(t) and DataVolumesInDwQueue(t) is the sum of submitted data volumes that are ready to be uploaded and downloaded respectively.

5.2 Workflow Operations

In our approach, the workflow of the submitted MapReduce job is not ready to be directly scheduled into the hybrid cloud resources becuase data uploading and downloading shall be reflected in the workflow and additional meta data of the submitted MapReduce job is required, especially the expected data sizes resulted by map and reduce operations. According to that, multiple workflow operations are required.

5.2.1 Generating workflows

When a user submits a MapReduce job, the system shall generate the corresponding workflow as DAG. In this case, the user must supply additional meta data like the number of mappers and reducers, expected computation costs for each mapper and reducer (number of interactions) and the expected transferred data volumes. Fig.2 shows MapReduce job as a workflow.



MapReduce job metadata	
X	# of map function instructions
Y	# of reduce function instructions
Z	# of results accumulation instructions
A	Map function key-value output size
B	Reduce function key-value output size

Figure 2: MapReduce job as workflow.

To simplify later explanations, let us assume the use case of scheduling the two MapReduce jobs shown in fig.3, where R1 and R2 tasks will be allocated later in the public cloud and the rest are allocated in the private one.

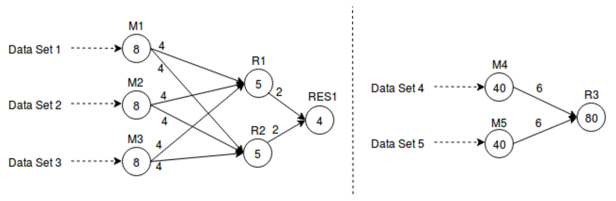


Figure 3: The usecase submitted MapReduce jobs.

5.2.2 Tasks Injection

In some cases, additional tasks shall be added to the workflow like:

- Entry and exit tasks are used to combine multiple separated workflows in a single one. Also, they are useful when a workflow has multiple entry and exit tasks as in case of MapReduce workflow. The computation and data transfer costs for both tasks equal zero. Fig.4 shows how both submitted MapReduce jobs are combined.

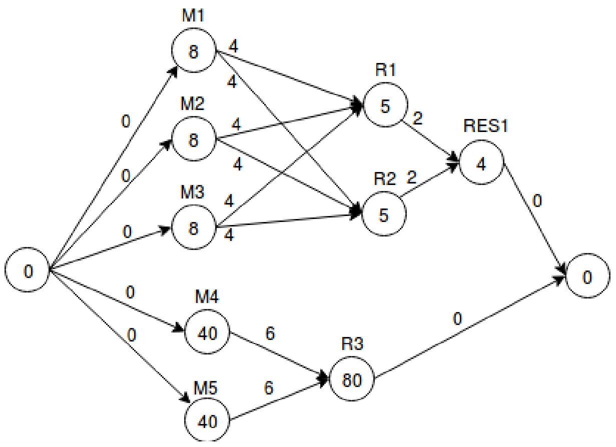


Figure 4: Combining multiple workflows.

- Data uploading task shall be injected before each public cloud allocated task when its parent is allocated in the private cloud. The same uploading tasks shall be also injected after each task allocated in the private cloud when its child is allocated in the public one, while uploading data consumes both clouds resources. The uploaded data will not be exchanged locally anymore. Accordingly, in case of uploading data, local data transfer costs in the cluster will be zero. The time that the processors take for uploading data v from a processor p_i in the private cloud to a processor p_j in the public cloud is defined as:

$$upt_{ij}(v) = \max\left\{pdtt_i(v), pdtt_j(v), \left(\frac{v}{UP}\right)\right\} \quad (7)$$

- Data downloading task shall be injected after each public cloud allocated task when its child is allocated in the private cloud. As in data uploading tasks, the same downloading tasks shall be also injected before each task allocated in the private cloud when its parent is allocated in the public one, while downloading data consumes both clouds resources. The downloaded data will not be exchanged locally anymore .Accordingly, in case of downloading data, local data transfer costs in the cluster will be zero. The time that the processors take for downloading data v from a processor p_i in the private cloud to a processor p_j in the public cloud is defined as:

$$dwt_{ij}(v) = \max\left\{pdtt_i(v), pdtt_j(v), \left(\frac{v}{DW}\right)\right\} \quad (8)$$

5.2.3 Merge

Allocating multiple data dependent tasks into one processor reduces data transfer costs between processors. Hence, multiple workflow tasks can be merged into one. There are three merging types:

- Vertical merge: Merging multiple tasks belong to the same data dependency level into one. The computation cost (number of instructions) of the resulting task equals the sum of computation costs of all merged tasks. Also, data dependency links shall be merged, there are two scenarios for merging data links:
  - If the transferred data volumes are identical then transferring data is performed only once. Fig.5.a shows an example of this case.
  - Otherwise, the transferred data volume in the resulting data link will be the sum of all data volumes of merged data links. Fig.5.b shows an example of this case.
- Horizontal merge: Merging sequential tasks into one. The data transfer costs in this case equal zero. Fig.6 shows an example of horizontal merge.
- Nested merge: Combine the vertical merge and the horizontal one, e.g. merging the vertically merged mappers with the vertically merged reducers horizontally.

After merging tasks, some entry and exit pre-injected tasks might be deleted.

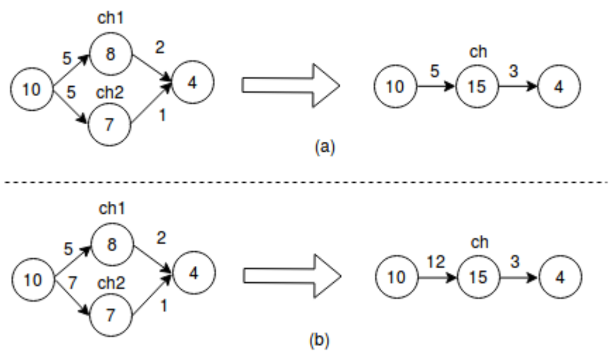


Figure 5: Vertical merge of tasks.

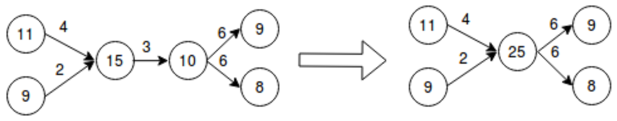


Figure 6: Horizontal merge of tasks.

5.2.4 Rescaling Values

Before executing the workflows on the simulation tool, rescaling of values is needed to accelerate the simulation execution. This is done by dividing tasks execution cost values on their greatest common denominator. Also, the same rescaling process is done for data volumes. In addition, reducing time units, if possible, speeds up the simulation stage. For example, if all values of tasks are in hours we can consider them in seconds.

5.3 Architecture

The main objective of our scheduling approach is to reduce monetary costs and meet deadlines. The scheduling process is divided into multiple layers:

1. High-level scheduling: Its main objective is to decide for each task whether to be executed on the private cloud or on the public one. Task t_i has a public cloud cost ratio which is defined as

$$pcr(i) = \frac{in_i}{e_{ai} + e_{ib}} \quad (9)$$

   Where t_a and t_b are a parent and a child tasks of t_i respectively. The high-level scheduler selects tasks with high public cloud ratio first to be allocated in the public cloud.

2. Intermediate-level scheduling: This layer consists of two schedulers, The private cloud workflow scheduler, which its main objective is to meet deadlines constraints. The second is the public cloud workflow scheduler, which its main objective is to minimize monetary costs and meet deadlines constraints.

3. Simulation: Executing the allocation plans provided by both intermediate schedulers and then sending an execution report to be evaluated. This report mainly contains the private cloud plan, which includes the execution schedule of all tasks allocated in the private cloud, and the public cloud plan, which includes the

execution schedule of all tasks allocated in the public cloud with resources reservation schedule.

4. Overlapping: Overlapping the private and public simulation execution reports into a comprehensive one and update the absolute execution time of each task into a relative one that represents the real execution starting time of a task.

5. Evaluation: Evaluating the execution report and provide a feedback to the rescheduling process. Depending on this feedback, the high-level scheduler changes the allocation plans. There are four feedback options: The first feedback option is that both schedules are not visible, in this case meeting both budget and deadlines constraints is not possible and scheduling process will stop. The second option is that the public cloud schedule is visible but the private one is not, this means that the deadline for one or more private cloud allocated tasks is exceeded and more tasks shall be transferred to the public cloud. The third option is that the private cloud schedule is visible and the public one is not, this means that the budget is exceeded and the tasks that are allocated in the public cloud shall be transferred to the private one. The last option is that both schedules are visible, in this case the allocation plan will be compared with the best plan so far. If it is better, then it will be submitted. Also, more iterations might be performed to optimize the allocation plan.

5.3.1 *System Flow Chart*

Firstly, the workflows will be generated and combined into one by injecting entry and exit tasks. The high-level scheduler decides to schedule R1 and R2, as an example, in the public cloud and the rest in the private one. The private and public allocation plans contain injected uploading and downloading tasks. Each data uploading operation costs both clouds, the private one while sending, and also the public one while receiving.

These intermediate schedulers are considered as a black box at the moment, many of the state of the art implemented schedulers can be applied. After simulating both allocation plans, the execution reports of both simulations, which contain the absolute execution time of each task on its hosting processor, will be sent to the overlapping stage. In this stage, the real execution time of tasks will be calculated e.g. if M1, M2 and M3 took 50 minutes and uploading data to the public cloud took 5 minutes, then the R1 and R2 will be started at minute 55, and so on.

After that, the final execution report will be sent to the evaluator to check if all constraints are met or not. If yes and the allocation plan cost is cheaper than the previous optimal cost plan then it will be submitted, otherwise it will be skipped.

Before starting a new iteration, the results must be logged. Hence, the high-level scheduler does reallocation of tasks with respect to historical logged data. Fig.7 shows the system components and detailed flowchart.

6. CONCLUSION

In this paper, we have proposed a model and an architecture of static multi-tier workflow scheduling in hybrid cloud. The presented model supports MapReduce jobs and takes into account data transferring costs on processors and sharing limited internet bandwidth among clouds' processors. The selected scheduling approach is a meta-heuristic one that runs in multiple layers and iterations to meet deadlines and minimize monetary costs.

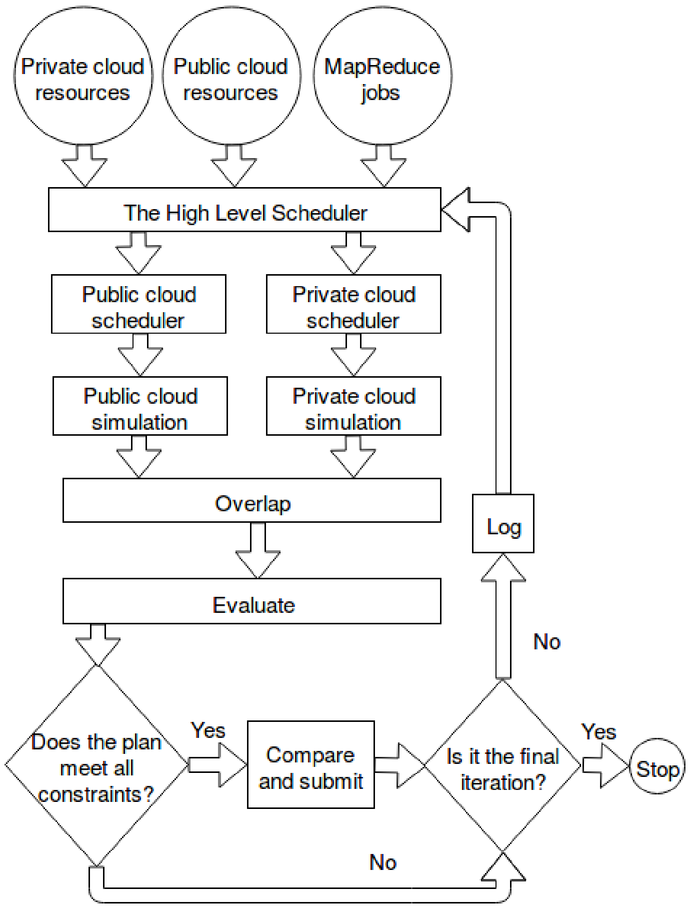


Figure 7: The System Flowchart.

7. REFERENCES

[1] Ye, Xin, et al. *"A Survey on Scheduling Workflows in Cloud Environment."* Network and Information Systems for Computers (ICNISC), 2015 International Conference on. IEEE, 2015.

[2] Mohamed, Ehab, and Zheng Hong. *"Hadoop-MapReduce Job Scheduling Algorithms Survey."* Cloud Computing and Big Data (CCBD), 2016 7th International Conference on. IEEE, 2016.

[3] Bittencourt, Luiz Fernando, and Edmundo Roberto Mauro Madeira. *"HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds."* Journal of Internet Services and Applications 2.3 (2011): 207-227.

[4] Topcuoglu, Haluk, Salim Hariri, and Min-you Wu. *"Performance-effective and low-complexity task scheduling for heterogeneous computing."* IEEE transactions on parallel and distributed systems 13.3 (2002): 260-274.

[5] Singh, Khushboo, Mahfooz Alam, and Sushil Kumar Sharma. *"A survey of static scheduling algorithm for distributed computing system."* International Journal of Computer Applications 129.2 (2015).