VOL. 10, NO. 7, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

# FAULT TOLERANT BASED HYPER-HEURISTIC ALGORITHM FOR TASK SCHEDULING IN CLOUD

R. Priyanka, P. Priyadharsini and M. Nakkeeran
Department of Computer Science, India
E-Mail: preethi.me13@gmail.com

**ABSTRACT**

Cloud computing is one of the recent emerging technologies for providing classy services by means of the Internet based on the requirements of the users. Numerous efficient scheduling algorithms are required to make a valuable use of terrific capabilities of the cloud environment. The aim of task scheduling is to schedule the tasks within the given deadline to achieve minimum makespan. The heuristics scheduling algorithm schedules the tasks on cloud by multiple iterations. The diversity revealing and improvement revealing operators are used to determine which low-level heuristic is to be used for finding enhanced solutions in task scheduling. The heuristics task scheduling should be fault tolerant to overcome the failure of tasks. The proposed Fault Tolerant Based Hyper Heuristic Algorithm (FT-HHA) provides fault tolerance in task scheduling by using task replication and task resubmission. FT-HHA schedules the tasks within the given deadline even in the occurrence of failures. The experiments were carried out in a simulated cloud computing environment by scheduling tasks in the existence of malfunction which are generated randomly.

**Keywords:** cloud computing, diversity revealing fault tolerance, heuristics scheduling, improvement revealing.

## 1. NTRODUCTION

Cloud computing is the way of using inaccessible servers on the internet to handle, store and process data instead of using a workstation. Cloud computing is better hooked on to three categories: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service). Iaas provides servers and storage on demand with the consumer paying accordingly. Paas allows the users to build and develop the applications within a providers framework. Saas enables the customers to use an application on demand through the browser.

Cloud computing allow the users to access the applications and data from any computer from any location at any time because they are stored on a remote server. It trim down the need for companies to acquire top of the line servers and hardware or engage users to run them since it is all maintained by a third party. There is no need for purchasing any software licenses for every user because the software servers are stored and executed remotely by the cloud. By means of centralizing bandwidth, storage, processing and memory in an offsite environment for a charge, cloud computing can significantly minimize the costs. The cloud can also store data therefore companies do not have to residence servers and databases themselves.

In cloud computing, scheduling is the progression of captivating decisions regarding the allocation of available capacity and/or resources to jobs and/or customers on time. Millions of user share cloud services by submitting their millions of computing task to the cloud computing environment. Scheduling of these millions of task is a clash to the cloud environment Scheduling process in cloud is divided into three stages namely; Resource discovering and filtering, Resource selection, Task allocation. In Resource discovering and filtering the datacenter broker discovers the resources present in the network system and collects status information about the resources. In Resource selection the target resource is selected based on the requirements of task and resource. This is a deciding stage. In task allocation, the task is allocated to selected resource.

## 2. JOB SCHEDULING IN CLOUD

The needs of job scheduling in cloud computing are load balance, quality of service, economic principles, best running time, throughput. With computing systems being shifted to cloud-based systems progressively, one of the main characteristics is that it works on a pay-as-you-use basis. Several studies attempted to define the scheduling problem on cloud systems as the workflow problem, which can be further classified into two levels: service-level (platform layer and static scheduling) and task-level (unified resource layer and dynamic scheduling). Different from grid computing, the user can install their programs on the virtual machines (VMs) and determine how to execute their programs on the cloud computing system. For these reasons, although both grid computing and cloud computing are heterogeneous, the key issues they face are very different. A good example is the cost and latency of data transfer on these environments. That is why some studies added more considerations to their definitions of scheduling on cloud. For instance, a couple of studies, used directed acyclic graph (DAG) to define the scheduling problem on cloud. The basic idea is to use the vertices of a DAG to represent a set of tasks and the edges between the vertices to represent the dependencies between the tasks.

## 3. HYPER HEURISTICS

Hyper-heuristics aim to discover some algorithms that are capable of solving a whole range of problems, with little or non-direct human control. Heuristic techniques are often referred to as “search algorithms”. The problems are solved by discovering a solution from

the set of all possible solutions for a given problem, which is regarded as the "search space"[1]. Non-deterministic search techniques such as simulated annealing method, local search methods, evolutionary algorithm and other search algorithms offer an alternative approach to an exhaustive search to solve complicated computational problems within a sensible amount of time. These methods guarantee for finding a solution at any time, but it may not be optimum.

Hyper-heuristics must positively influence the selection of heuristics. The optimized heuristics for a given problem should compute high quality solutions. The learning point should refine the algorithms, so that the algorithm solutions subsequently meet the needs of the training set and problems of a certain class can be solved more efficiently. The response mechanism should move towards optimum algorithm solutions in the workspace, as it guides the selection of heuristic. The Algorithm Selection Problem represents in a three-dimensional coordinate system namely the relationship between a problem instance, an algorithm solution and its performance. Comparatively, the two-level model offers a clear separation between the optimization of an algorithm and the optimization process of a specific problem.

## 4. RELATED WORK

The basic idea of heuristics is to use three key operators—transition, evaluation, and determination [2]—to "search" for the possible solutions on the convergence process. $t$ denotes the iteration number; $t_{max}$ the maximum number of iterations or the stop criteria. More precisely, the transition operator creates the solution s, by using methods which could be either perturbative or constructive or both; the evaluation operator measures the fitness of s by using a predefined measurement; and then the determination operator determines the next search directions based on the s from the transition operator and the evaluation operator.

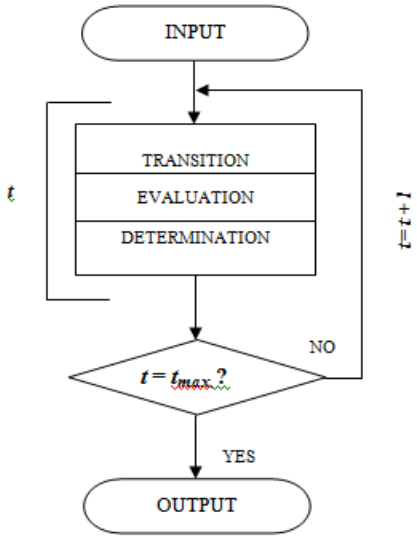



**Figure-1.** Outline of Heuristics.

The basic idea of [5] hybrid-heuristic algorithm is to combine heuristic algorithms to perform the transition (T), evaluation (E), and determination (D) at each iteration, where $H_i$ denotes one of the heuristic algorithms. This kind of integration may compensate for the intrinsic weak points of specific heuristic algorithms. A critical problem is that although the hybrid-heuristic may have a higher chance to find a better result than a single heuristic does, it generally takes a longer computation time than heuristics at each iteration of the convergence process.

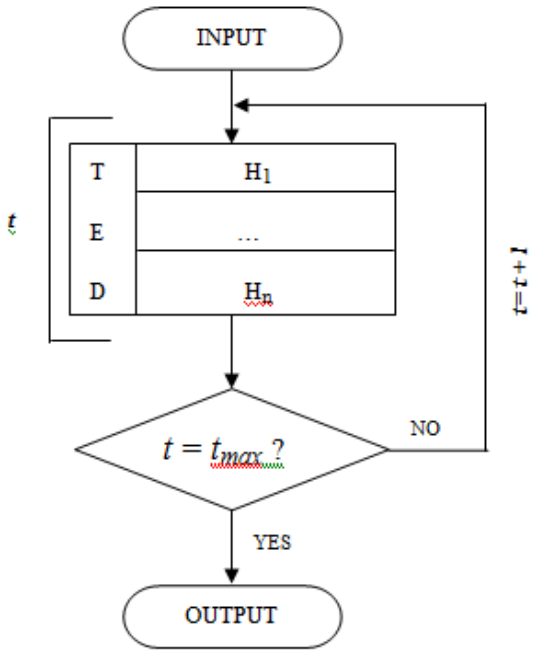



**Figure-2.** Outline of Hybrid Heuristics.

They are simple and easy to implement. Some rule based deterministic algorithms can find acceptable solutions quickly. Most of them are compatible to each other, so some studies have tried to integrate two or more non-metaheuristic algorithms to solve the scheduling. The most algorithms, such as the well-known branch-and-bound and dynamic programming, are normally time consuming because the number of checks they have to perform is very large. The deterministic algorithms, while they are very fast and easy to implement, they are easily falling into local optima. The results obtained by these algorithms may be far from optimal or even acceptable.

## 5. THE PROPOSED ALGORITHM

### a) Fault tolerant based Hyper Heuristic scheduling algorithm

A high-performance hyper-heuristic algorithm is proposed for scheduling the jobs on cloud computing systems to shrink the makespan. From the pool of candidate heuristics, one of the heuristic algorithms will be picked by the low-level heuristic (LLH) selection operator as the heuristic algorithm that is to be performed. Two


revealing operators one for diversity revealing and one for enhancement revealing are proposed for the proposed algorithm to regulate the effectiveness to employ the low-level heuristic algorithm. The suggested algorithm can be useful to both sequence-dependent and sequence-independent scheduling problems.

The proposed algorithm use the multiplicity revealing and enhancement revealing operators to balance the escalation and diversification in the search of the solutions during the convergence process. As far as the proposed algorithm described herein is concerned, the low-level heuristic candidate pool consists of particle swarm optimization, genetic algorithm, ant colony optimization and simulated annealing. The selected enhanced hyper-heuristic algorithm will then be performed repeatedly until the termination criterion is met. More specifically, the selected LLH will evolve the solution for iterations by using the determine function to balance the escalation and diversification of the search directions, which in turn rely on the information provided by the enhancement detection operator.

**b) The improvement revealing operator**

A simple random method is used to select the low-level heuristic $H_i$ from the candidate pool H. According to the observation, the best so far makespan (BSFMK) for both SA and GA could continue to improve the results at early iterations (e.g., less than 250 iterations), but it is difficult to improve the results at later iterations (e.g., after 800 iterations), especially when the search directions converge to a small number of directions. If the selected $H_i$ cannot improve the BSFMK after a row of $\phi_{ni}$ iterations, the improvement revealing operator will return a false value to the high level hyper center to indicate that it should pick up a new LLH. The improvement revealing operator will return a false value in three cases: the maximum number of iterations $\phi_{max}$ is reached, the number of iterations $\phi_{ni}$ the solutions are not improved is reached, and when the stop condition is reached.

**c) The diversity revealing operator**

In addition to the improvement revealing operator, the diversity revealing operator is used by HHSA to decide "when" to change the low-level heuristic algorithm $H_i$. The diversity of the initial solution $D(A_0)$ will be used as a threshold $\Theta$, i.e., $\Theta = D(A_0)$. The diversity of the current solution $D(AZ)$ is computed as the average of the task distances between individual solutions. If a task in two different individuals is assigned to the same VM, the task distance is 0; otherwise, the task distance is 1. If the diversity of the current solution $D(A)$ is less than the threshold $\Theta$ (the diversity of the initial solution), this operator will return false, and algorithm will then randomly select a new LLH.

**Algorithm**

[1] Set up parameters.
[2] Input the scheduling problem.
[3] Initialize the population of solutions $A = \{a_1, a_2, ..., a_N\}$.
[4] Randomly select a heuristic algorithm $H_i$ from the candidate pool H.
[5] While the termination criterion is not met
  a. Update the population of solutions $A$ by using the selected algorithm $H_i$.
  b. $F_1$=improvement_revealing(A).
  c. $F_2$=diversity_revealing(A).
  d. If $\Psi(H_i, F_1, F_2)$
  i. Randomly select a new $H_i$.
  ii. Schedule(t, $H_i$)
  e. End.
[6] End.
[7] Output the best so far solution as the final solution

**d) Preprocessing module (PM)**

The main functionality of preprocessing module is to calculate all the parameters like threshold, heuristic metric, deadline of each task etc., which are required in the process of scheduling tasks. The PM accepts the data required for the task from the user in the form of an abstract data structure template.

**e) Replication based scheduling module (RSM)**

RSM module sorts all tasks in the ready queue as in QWS algorithm based on Instructions time ratio and Number of services. Instruction time ratio is the ratio between the deadline of the task and the number of instructions in the task. The tasks with less instruction time ratio are scheduled first. If a task requires further number of services, it may block the tasks which require less services and thereby increasing the waiting time of these tasks. So the task with less service is scheduled first. After sorting all the tasks in the ready queue, it checks whether the task requires any additional computation services or storage services. If the task is allied to storage service, then the task is mapped to the storage server.

The tasks in ready queue are mapped with the available number of services in the datacenter by means of the information that is present in registry. If any of the services that are available in a datacenter are busy, then it will check for the other datacenters and then it assigns to the next server that is free. If all datacenters are busy, then it will be mapped to the wait queue of the datacenter which has fewer load when compared to others.

**f) Resubmission based executor module(REM)**

` REM sends all mapped tasks to the relevant datacenter and also it has to wait for the acceptance and reply from the datacenter. The datacenter can allow the task or decline the task based on excess information. If the datacenter accepts the task, then REM sends the task to the datacenter by assigning a sole version identity to the task and then waits for the result.

While sending the task it also starts a timer by spawning the thread with the expected time to execute the

task. REM waits till the timer expiry or till the result comes back. Once REM gets the result, it checks for the correctness of a task by checking the given version id and the obtained version id. If the completed task is correct then, it sends the signal to the node where the task was executed to update the datasets in data store and also to stop all the other replicas. REM sends the result to preprocessor which masks the dependencies of all its child tasks.

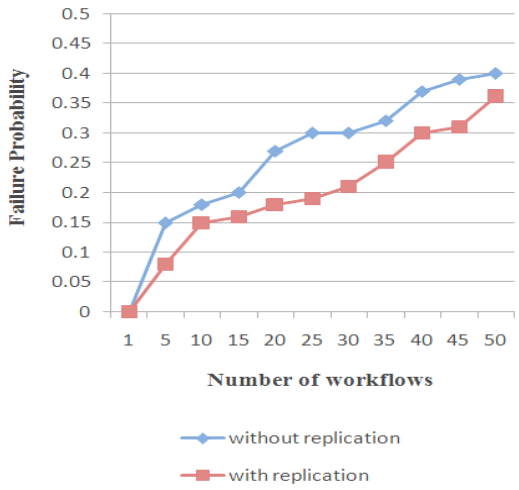
**g) Task scheduler(TS)**

Tasks may be replicated in different nodes so all those nodes may access the similar set of datasets. Consequently the datasets are replicated. The management of these data sets is done by the TS. The strategy is to maintain two copies of the datasets namely local copy and primary copy. After getting the datasets from the data store to the node the copy of datasets in node becomes as a local copy. All the changes done by the node are modified locally. Once the signal comes from the REM, the updated datasets are sent to the data store and also TS invalidates all the local copies in other nodes. TS maintain a table to keep track of all the replica information which helps in invalidating the datasets.

The algorithm for scheduling and mapping tasks on services is shown below:

```
void schedule (t,Hi )
{
        while(  ready_queue not empty )
        {
                t = first task in ready_queue
                s = getservice(t)
                send task to the data center by placing in
                run queue
                dequeue(ready_queue)
                make s as busy
        }
}
int getservice(t)
{
        Select service s such that
                Executiontime(t) < deadline(t)
        return s
}
```

**h) Failure probability prediction**




## 6. CONCLUSIONS

The proposed algorithm uses two revealing operators on impulse to define when to variant the low-level heuristic algorithm. The can not only deliver better results than the traditional rule-based scheduling algorithms, it furthermore overtakes the other heuristic scheduling algorithms, in resolving the workflow scheduling and map-task scheduling difficulties on cloud computing environments. The scope is to understand the type of job failures with the intention of improving the dependability of the essential cloud infrastructure from the perspective of cloud providers. Further, the focus is to explore the potential for failure prophecy and incongruity revealing in cloud applications in order to avoid wastage of resources by jobs that fail in due course.

## REFERENCES

[1] C. W. Tsai and J. Rodrigues. 2014. “Metaheuristic scheduling for cloud: A survey”, IEEE Systems Journal , vol. 8, no. 1, pp. 279–297.

[2] Chun-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang and ChuSing Yang. 2014. “A Hyper-Heuristic Scheduling Algorithm for Cloud”, IEEE Transactions on Cloud Computing, January.

[3] S. Abrishami and M. Naghibzadeh. 2012 “Deadline-Constrained Workflow Scheduling in Software as a Service Cloud”, Scientia Iranica, Volume 19, Issue 3, June.

[4] Nanduri. R, Maheshwari. N, Reddyraja. A and arma. V. 2012, “Job Aware Scheduling Algorithm for MapReduce Framework”, Cloud Computing Technology and Science (CloudCom), IEEE Third International Conference.


[5] M. Rahman, X. Li and H. Palit. 2011. "Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment," Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Workshops, pp. 966–974.

[6] Bala and I. Chana. 2011. "A survey of various workflow scheduling algorithms in cloud environment," Proceedings of the National Conference on Information and Communication Technology, pp. 26–30.

[7] Suraj Pandey, LinlinWu, Siddeswara Mayura Guru and Rajkumar Buyya. 2010. "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", Advanced Information Networking and Applications (AINA), 24th IEEE International Conference.

[8] Mohsen Amini Salehi and Rajkumar Buyya. 2010. "Adapting Market-Oriented Scheduling Policies for Cloud Computing", Algorithms and Architectures for Parallel Processing, Springer, pp 351-362.

[9] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker and Ion Stoica. 2010. "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", Proceedings of the 5th European conference on Computer systems, ACM.

[10] D. Laha and U. Chakraborty. 2009. "An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling", The International Journal of Advanced Manufacturing Technology, vol. 44, no. 5-6, pp. 559–569.