DISTRIBUTED HEURISTIC LOAD BALANCING SYSTEM USING REPLICATION APPROACH

[1]D.Ragunath, [2]Dr.V.Venkatesa Kumar, [3] Dr.M.Newlin Rajkumar
[1]Research Scholar, Computer Science & Engineering, Anna University Regional Centre, Coimbatore, Tamil Nadu
[2]Assistant Professor & Head ,Computer Science, Anna University Regional Centre, Coimbatore, Tamil Nadu
[3]Assistant Professor, Computer Science, Anna University Regional Centre, Coimbatore, Tamil Nadu

Abstract: Cloud computing is a promising and up-coming technology which allows the users to store, share their data and provides different types of services. Cloud computing provides an expert and an enhanced way to perform the works which were submitted by the users in various characteristics. To improve the performance, there are several rule based algorithms are widely used in cloud computing environment. Due to its huge size nature, selection of algorithm became very tedious. Existing schedulers often incur a high communication overhead when collecting the data required making scheduling decisions, hence delaying job requests on their way to the executing servers. A novel scheme is proposed in current system that incurs no communication overhead between the users and the servers upon job arrival consequently removing any scheduling overhead from the jobs critical path. A new high-performance improved **hyper-heuristic algorithm** is proposed for scheduling on distributed systems to reduce the makespan problem (makespan is referred as the time difference between the start and finish of a sequence of jobs or tasks) and it also introduce a new scheme, which considers the job failure at the time of scheduling.

Keywords: Cloud Computing, Load Balancing, Replication, Heuristic approach, Job scheduling, Job allocation, Make span.

1. INTRODUCTION

Cloud computing is a best solution for the delivery of computing as a service rather than a product, whereby software, information, and shared resources are allowed to computers and other devices as a utility over a network. Cloud computing is better captivated on to three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) [1]. Iaas provides servers and storage on demand as pay for use. Paas allows the users to frame and establish the applications within a provider's scheme. Saas enables the clients to use an application on demand through the web 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. Cloud environment can consider different parameters such as bandwidth, memory, storage and other resources based on the user requirement. This facility provides a cost effective process to the cloud users. For this process, cloud service providers charging like pay-as-you-click and pay-as-you-use‖ functions.

The cloud can also store data therefore users and companies do not have to own servers and databases for different services. 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, which are Resource selection, discovering and filtering and Task allocation. In Resource discovering and filtering the data center broker

discovers the resources present in the network system and collects status information about the resources [2]. In Resource selection the target resource is selected based on the requirements of task0020and resource. This is a deciding stage. In task allocation, the job is allocated to selected resource from the set of resources. The job scheduling has tremendous concentration due to its vast use and advantage, these include load balance, quality of service checking and improvement, cost adaptive frameworks, and optimal time based allocation and 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).

Unlike grid computing, the cloud environment offers both resource sharing and virtual data access in a single and heterogeneous environment. This allows the user to install and work their programs on the virtual machines. 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 a-cyclic graph (DAG) [3] to define the scheduling problem on cloud. The basic idea is to use the vertices of a DAG to represent set of tasks and the edges between the vertices to represent the dependencies between the tasks.

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‖[4]. 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 authors in the paper [5], suggests an algorithm solution, which defines the relationship between multi-dimensional coordinate systems. The multi-level model provided in the paper is an optimized solution for a specific job scheduling problem.

2. RELATED WORK

The basic idea of heuristics is to use three key operators transition, evaluation, and determination – to search‖ for the possible solutions on the convergence process. t denotes the iteration number; tmax the maximum number of iterations or the stop criteria. More precisely, the transition operator creates the solution K, by using methods which could be either perturbation or constructive or both sometimes. The evaluation operator measures the fitness of K by using a pre-defined measurement and then the determination operator determines the next search directions based on the s from the transition operator and the evaluation operator.

The basic idea proposed in paper [6] is hybrid heuristic algorithm is to combine heuristic algorithms and perform the scheduling process. In this paper, the Transition is mentioned as (t) evaluation (e) and determination (d) at each iteration of the process. In this process, the transition t denotes the initial parameter. This kind of integration may recompense for the intrinsic weak points of specific heuristic algorithm in the specific work. The main problem in the proposed algorithm in the paper has a higher chance to find a better result than a single heuristic method in the solutions. Another problem of this method is that needs longer computation time than heuristics in every iteration of the convergence process.

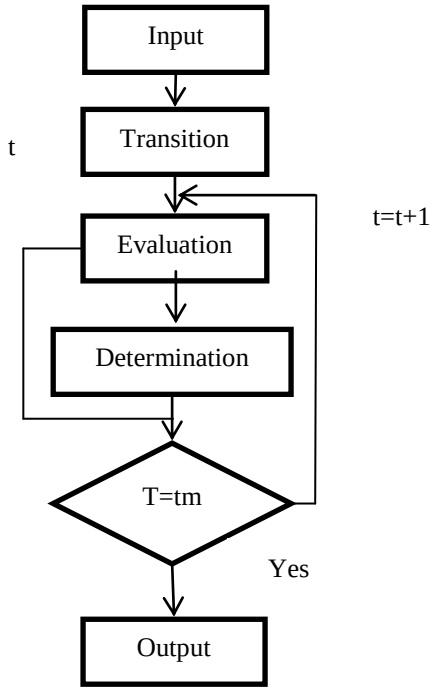

Figure 1: Heuristics

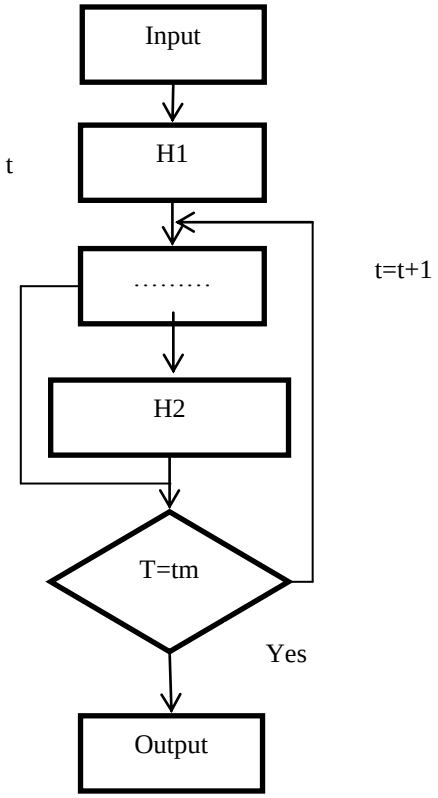

Figure 2: Hybrid Heuristics

The hybrid heuristics are simple and easy to implement. Several rule based deterministic algorithms can find acceptable solutions rapidly those solutions can integrate with one another. This feature motivates some studies to integrate two or more non Meta heuristic algorithms to solve the scheduling issues. The popular and well-known branch and bound and dynamic programming algorithm, which comes under meta-heuristic algorithm list usually time-consuming due to its multi-check scenario. Unlike the branch and bound, the deterministic algorithms are very fast and easy to implement. This is suitable for all scenarios and it can easily fall into local optima. The results obtained by these algorithms are not optimal or even acceptable.

3. PROBLEM DEFINITION

Cloud load balancing and replica allocation became challenging when the load is dynamic and that is not predictable. The VM provisioning schemes are extended to the high resource infrastructures and the replication made as per the demand. But only few methods are used dynamic replication schemes for effective load handling, however the schemes are not considered the customized user perception based VM allocation. In the proposed system we aims to provide a new customized parameter based approach for VM provisioning and replication for load handling.

4. PROPOSED SYSTEM

In the proposed system, we provided a new improved hyper heuristic approach to reduce the make-span and load allocation problem. The proposed system reduces communication overhead between the users and the servers upon task input. So the proposed system aims at removing scheduling overhead from the server side. A new high-performance improved hyper-heuristic algorithm is proposed for job scheduling on distributed systems to reduce the make-span and considers the job failure at the time of scheduling. The proposed system has the following steps in implementation.

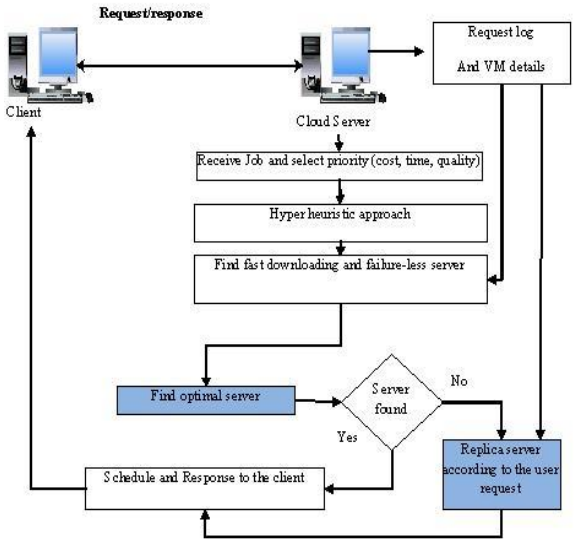

Figure 3: Architecture of the proposed system

The fig 3.0 represents the overall architecture of the proposed system, which selects and allocates optimal server when the client request has made.

4.1 Cloud Infrastructure Creation: Cloud computing provides virtual services. Cloud infrastructure model allows users to develop a new architecture where a dedicated resource platform runs for hosting base service workload, and a separate and shared resource platform serves in multi-server peak load. The infrastructure based on socket programming to provide the multi user and also a multi-server application platform.

4.2 VM provisioning And Allocation: The flow fairness algorithm collects the requests from the users expressed as types and determines the allocation by calling the allocation process.. Once the allocation is determined, the mechanism provisions the required number and types of VM instances. Then, the mechanism determines the payments by calculating the user preference function. The users are then charged the amount determined by the mechanism.

4.3 Service priority Selection: The service priorities like optimal, costlier, speed will be selected by the users. Based on the selection the request will be handled by the different virtual machines. Each user request is having a deadline to respond. The VM selection process should complete within the deadline.

4.4 Hyper Heuristic approach: Hyper Heuristic planner has two heuristics to reduce the searching space. First of all, it uses main schemes and auxiliary schemes alternatively during the optimization process. Second of all, the planner uses the cost model to prune the ―unpromising‖ transformations. Third, the planner is rule-based. The rule is defined to consist of two components: condition and action, where the condition is usually defined based on the cost and performance optimization goal (e.g., the estimated monetary cost can be reduced by 20 percent) and the action consists of transformations on the workflow.

4.5 Cost: The flow fairness algorithm does not assume any specific initial instance assignment. We present a number of heuristic based methods for initial instance assignment. If the priority is a costlier machine then the cost should be bear by the user (costlier in the sense like extra bandwidth, speed and memory etc.,).

4.6 **Time:** Application Service period is an important tuning parameter in the VM schedule planner. If the period is long, more workflows are buffered in the queue. When the period parameter is short, theoptimization space gets smaller and the chance for operating transformations to reduce cost decreases.

4.7 Optimal: (Quality) To make the optimization plan, more combinations of tasks need to be checked for transformations and the optimization space becomes much larger.

4.8 Replica allocation: Before sharing the load in the cloud, the system should verify the load of the virtual machines of the cloud. Allocating sub replica is more important, here the sub servers are the service provides for the client in the cloud when the main cloud load exceeds. So data availability is more considered in this module. The system finds the base crowd and flash crowd using the server threshold and the past behavior. Before data transmission the servers should be

represented by their unique properties such as id, name and other details.

5. IMPROVED HYPER HEURISTIC ALGORITHM

The algorithm predicts the workloads and tasks of each virtual machine at the time client request. Based on that, the system identifies a threshold. The proposed system follows the following process:

i. Prediction of resources.
ii. Analyzing Pending tasks
iii. Past behavior (historical list)
iv. Current top k values

Here work load prediction algorithm has been applied. Firstly, most workload predictions are based on history record analysis. And enough data are needed for the accuracy of workload prediction. Therefore, if the application is very new and there are not enough records for prediction, workload prediction will be difficult. So this module enables the collection of data flows and predicts the workload. In propose framework, since it is unreasonable to determine a node's state through single monitoring, the state vector of a node is calculated by its historical monitoring states collected by workload monitor of the node. And currently, computation resource utilizations in the state vector are the averages of their recent historical monitoring values. If the state vector of a node is calculated, this uses the following function to determine the type of workload of the node.

$$L(n) = \begin{cases} \text{overloaded} & \exists i,\ u_i > \text{opt}_i \\ \text{underloaded} & \sum_{i=1}^{n} \mu_i (\text{opt}_i - u_i) > t \\ \text{normal} & \text{otherwise.} \end{cases}$$

In this function, ui represents the utilization of Resource i,μi represents the weight of Resource i, opt_i represents the defined ideal utilization of Resource i and t represents the threshold of under-loaded. Since excessive utilization of any resource can lead to poor system performance, the rule for determining overloaded is rational. And the rule for determining under-loaded takes all kinds of resource utilization into consideration as well as provides weight for elastic configuration. Therefore, it is rational, too.

The algorithm has the following features such as Total resources (TR), Total jobs(TJ) ,Execution speed of resources(Esp), Job length(JL), Population size(Ps), Maximum number of iteration(MaxI), Start time of job(Sj), End time of job (Ej) and Job failure count(Fcount)

EHHSA:

Step 1: initialize server parameters s_1, s_2, s_3... s_n

Step 2: receive client request C_R and find request type T.

Step3: if (T==||connection||)

Step 4: Then select s_1.

Step 5: else if (T==||download||) goto step 6

Step 6: find server queue Sq=calculateQueue(capacity-current_req)

Step 7: find fast_download_server Fs=Min (response_time- request time)

Step 8:if(Fs(Sq)>1)

Step 9: select Fs. Update the queue and data

Step 10: re-schedule

Algorithm1: Hyper Heuristic for load aware server selection

From the above algorithm, the system initiates the server parameters and initiates scheduling. The system has successfully demonstrated with different clients and 3 servers.

6. PERFORMANCE EVALUATION

The HHSA is implemented and demonstrated the cloud load analysis and allocation process with the following parameters.

Parameters: the implementation is carried out with one main server and two sub servers. There are 10 to 20 clients are created for experiment. After the implementation the performance is evaluated and compared with the existing algorithm.

Table 1: Download Time comparison table

Parameters	Existing (HHSA)	Proposed (Enhanced HHSA)
Download time(10 clients)	8	3
Download time(20 clients)	18	7

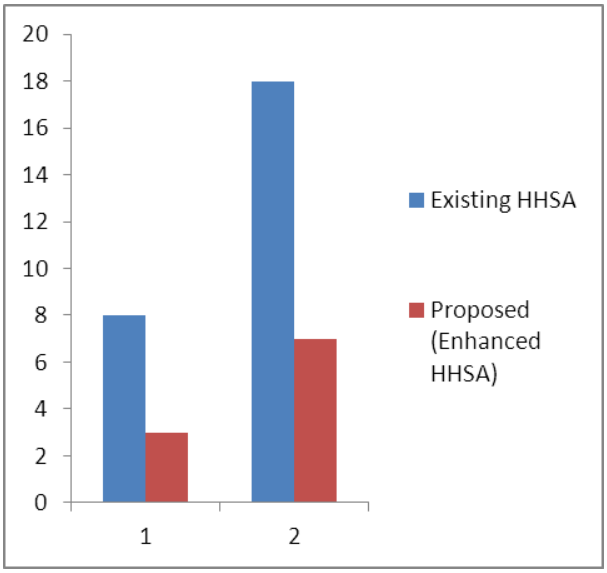

Figure 4: comparison chart

The fig 4.0 shows the comparison of the proposed system with the existing system. Due to the effective scheduling, the proposed system reduces download time for 10 to 20 clients.

6.1 Make span comparison:

The proposed system performs the makespan comparison which shows the proposed system took only 34 seconds for 10 clients.

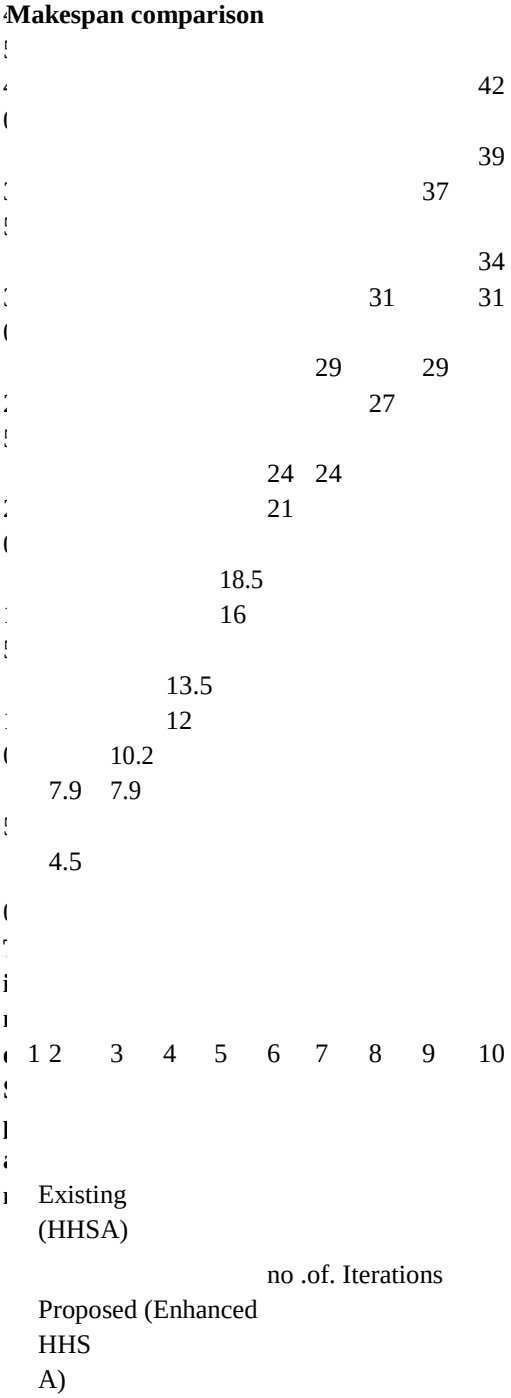

Figure 5: Comparison chart

The fig 5.0 shows the makespan comparison

between HHSA and EHHSA. The proposed system decreases the time delay for the allocation.

7. CONCLUSION

The proposed system creates a high-performance hyper-heuristic algorithm to find better scheduling solutions for cloud computing systems. The proposed algorithm uses replica based load balancing, which automatically determine when to change the low-level heuristic algorithm and a perturbation operator to fine-tune the solutions obtained by each low-level algorithm to further improve the scheduling results in terms of makespan. As the simulation results in C#.net the proposed algorithm can not only provide better results than the traditional rule-based scheduling algorithms, it also outperforms the other heuristic scheduling algorithms, in solving the workflow scheduling problems on cloud computing environments.

REFERENCES

[1] D. Breitgand, R. Cohen, A. Nahir, and D. Raz, On Cost-Aware Monitoring for Self-Adaptive Load-Sharing,‖ IEEE Journal on Selected Areas in Communication, vol. 28, no. 1, pp. 70–83, 2010.

[2] J. Dean and L. A. Barroso, ―The tail at scale,‖ Commun. ACM, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[3] S. Fischer, ―Distributed load balancing algorithm for adaptive channel allocation for cognitive radios,‖ in In Proc. of the 2nd Conf. on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom, 2007.

[4] B. Fu and Z. Tari, ―A dynamic load distribution strategy for systems under high task variation and heavy traffic,‖ in SAC ’03: Proceedings of the 2003 ACM symposium on Applied computing. New York, NY, USA: ACM, 2003, pp. 1031–1037.

[5] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, ―Autoscale: Dynamic, robust capacity management for multi-tier data centers,‖ ACM Trans. Comput. Syst., vol. 30, no. 4, p. 14, 2012.

[6] V. Gupta, M. Harchol Balter, K. Sigman, and W. Whitt, ―Analysis of join-the-shortest-queue routing for web server farms,‖ Perform. Eval., vol. 64, no. 9-12, pp. 1062–1081, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2007.06.012

[7] M. Harchol-Balter, ―Task assignment with unknown duration,‖ J. ACM, vol. 49, no. 2, pp. 260–288, 2002.

[8] B. Hayes, ―Cloud computing,‖ Commun. ACM, vol. 51, no. 7, pp. 9–11, 2008.

[9] IRCahce, http://www.ircache.net/, visited: 2013-10-26.

[10] M. Isard, ―Autopilot: automatic data center management,‖ SIGOPS Oper. Syst. Rev., vol. 41, no. 2, pp. 60–67, 2007.

[11] L. Kleinrock, Queueing Systems, Vol. I: Theory. Wiley Interscience, 1975.

[12] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. G. Greenberg, ―Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services,‖ Perform. Eval., vol. 68, no. 11, pp. 1056–1071, 2011.

[13] M. Mitzenmacher, ―How useful is old information?‖ IEEE Trans. Parallel Distrib. Syst., vol. 11, no. 1, pp. 6– 20, 2000.

[14] The power of two choices in randomized load balancing,‖ IEEE Trans. Parallel Distrib. Syst., vol. 12, no. 10, pp. 1094–1104, 2001.

[15] J. Moon and M. H. Kim, ―Dynamic load balancing method based on DNS for distributed web systems,‖ in E-Commerce and Web Technologies: 6th International Conference, EC-Web 2005, Copenhagen, Denmark, August 23-26, 2005, Proceedings, 2005, pp. 238–247.