iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** VII **Month of publication:** July 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Scheduling In Cloud Based On Hyper Heuristics

Bini T Bhaskaran[1], Sindhu S[2]
[1,2]*Dept. of Computer Science and Engineering,*
NSS College of Engineering, Palakkad, Kerala

Abstract*—Traditional scheduling algorithms are used for scheduling tasks in many cloud computing systems. They are used mainly because of their simplicity and easiness to implement. Heuristics when applied to these algorithms the performance increases. Hyper heuristics is used here. Genetic and simulated annealing algorithms are used in the candidate pool as low level heuristic algorithms. The performance of genetic algorithm is further increased by combining it with other evolutionary algorithm such as differential evolution. Comparisons based on maximum lateness, maximum tardiness, makespan and maximum flowtime show that the combined heuristic algorithm performs better than when each of these low level candidate solutions used individually.

Keywords*— Scheduling, Cloud computing, Heuristics, Evolutionary algorithms, Makespan, Maximum flowtime, Maximum tardiness.

I. INTRODUCTION

Many available solutions have been looked forward to enhance the performance of information systems mainly in terms of computation, analysis, and storage. Distributed computing as well as parallel computing was widely used to enhance the performance of a variety of information systems. The main issue is how to manage these computer resources efficiently. Among them, the most important one for the successful operation of computer system is scheduling [1]. Scheduling problems involve certain tasks that must be scheduled on some specific machines and it specifies when and on which machine each job is to be executed. Nowadays cloud computing is very popular. It provides on demand services or computer resources to the remote users over a network. It can also be considered as a pay per use model. It enables available, on demand network access to a shared pool of configurable computing resources that can be released and provisioned with minimum service provider interaction or management effort [2].

Virtual machines can be used to execute jobs in an on-demand manner. This new paradigm shifts the location of the computing infrastructure from the user site to the network thereby reducing the costs, both capital and management of hardware and software resources. And also the scheduling algorithms must minimize the delay and the production cost of executing a job. One of the main characteristics of cloud computing systems is that it works on a pay-as-you-use basis. Previously, the scheduling problem on cloud systems has been defined as a workflow problem. But nowadays it includes the reduction of cost also [9].

Many models of cloud computing have been used successfully on several information systems in recent years mainly due to the advancement of computer and internet technologies. Nowadays cloud computing systems provide a better way to carry out the submitted tasks in terms of responsiveness, scalability, and flexibility. Even though this is the case, most task scheduling problems on cloud computing systems are still difficult to solve. Most of the traditional scheduling algorithms are rule-based. scheduling algorithms widely used on today's cloud computing systems. These algorithms are simple and easy to implement.

But these rule-based scheduling algorithms are not at all appropriate for handling large-scale or complex scheduling problems. This is mainly because the results of these scheduling methods are far from the optimal.

Heuristics is applied to scheduling on cloud computing systems. For problem solving, heuristics refers to several experience-based techniques. For the purposes of learning and discovery it finds a solution which is not guaranteed to be optimal. But it is good enough for a given set of goals. When thorough search becomes impractical, heuristic methods are used to speed up the process of finding a satisfactory solution. The most important motivation for studying hyper-heuristics is to build systems which can handle classes of problems rather than solving an individual problem. There may be a large number of heuristics from which one can choose for solving a problem, and each heuristic has its own pros and cons. The idea is to automatically devise algorithms by combining the strength and compensating for the weakness of known heuristics. A typical framework includes a high level methodology and several low level heuristics [1]. When a problem instance is given, the high level method selects which low level heuristic should be applied and this depends on the search space of the problem and the current problem state. There exist a fundamental difference between meta heuristics and hyper-heuristics. Most of the implementations of meta heuristics search within the problem solution's search space. But hyper-heuristics always search within a search space of heuristics. So when using hyper heuristics, we are trying to find the right sequence or methods of heuristics suitable for a given situation rather than trying to solve a problem directly. That is, searching is done for a generally applicable methodology rather than solving a single problem instance [3].

©IJRASET 2015: All Rights are Reserved

User requests services from a datacenter broker. Datacenter broker represents a broker acting on behalf of the user. Broker determines which cloudlet to execute first and also in which virtual machine it runs. Suitable cloud service providers are also discovered by querying the Cloud Information Service (CIS). It hides the management of virtual machine, such as creation of virtual machine, submission of cloudlets to this virtual machines and also destruction of these virtual machines. Datacenter deals with processing of VM queries that is handling of VMs instead of processing Cloudlet-related queries [9].

II. PRIOR WORKS

A literature survey on various scheduling methods is given below. Various parameters have been used to compare various scheduling algorithms such as throughput, response time and resource utilization.

A. Earliest Due Date First

It is a dynamic scheduling method mainly used in many real time systems. Tasks are placed in a priority queue. Whenever an event comes for scheduling, the queue will be searched for the process closest to its deadline. Next this process is scheduled for execution. It can be considered as an optimal scheduling algorithm. This scheduling ensures all the jobs complete by their deadline. It guarantees that all deadlines are met. When the system becomes overloaded, to predict the processes that will miss the deadline becomes impossible. This is difficult to implement in hardware and also the representation of deadlines in different ranges becomes a tedious task [4].

B. First Come First Served

The first come, first served, commonly called FIFO scheduling algorithm is the simplest scheduling algorithm. It is not commonly used in most of the systems. It is sometimes used inside of other scheduling systems. The process which comes first(First In) is the first one to be dealt with (First Out). It is a non preemptive technique that is the process will run until it finishes. And is always implemented using a FIFO queuing. Short processes which are at the back of the queue have to wait for the long process at the front to finish, because non preemptive scheduling is used here. The waiting time always depends on the arrival order.

C. Shortest Job First

It is a scheduling method that selects the waiting process with the smallest execution time to execute next. It is non-preemptive in nature. The major advantage is that because of its simplicity it minimizes the average amount of time each process has to wait until its execution is complete. The major drawback is that process starvation occurs. When short processes are continuously added, process starvation occurs for processes which require a long time to complete. Another major drawback of this method is that the total execution time of a job must be known before execution. This can be effectively used with interactive processes. It can also be used in situations where accurate estimates of running time are available [3].

D. Priority Scheduling

Each process is assigned a priority. When equal priority processes come, they are scheduled in FCFS order. It is a special case of general priority scheduling. Priority can be defined externally or internally. Internally defined priorities use some measurable quantities or qualities to compute priority of a process. It includes time limits, memory requirements and file requirements. Those priorities defined externally are set by some external criteria such as the importance of process, or the type or amount of funds being paid for computer use. In this scheduling the scheduler selects tasks to run based on their priority as opposed to round robin. Priorities can be static or dynamic. At the time of creation, static priorities are assigned. Whereas based on the behaviour of each task, dynamic priorities are assigned. By using fixed priority pre-emptive scheduling, the scheduler ensures that, the processor executes the highest priority task of all those tasks that are currently ready to execute at any given time. Dynamic priority scheduling is a type of scheduling algorithm in which the priorities are calculated during the execution of the system.

E. Min Min Algorithm

It is a static task scheduling algorithm. Tasks are scheduled based on minimum completion time. Min-min task scheduling algorithm runs in polynomial time. At the same time, it produces efficient schedules and this algorithm has been evaluated for many different situations involving non-communicating or independent tasks. And also it does not guarantee a mapping which is optimal. Small tasks are chosen first for execution. The major disadvantage is that long tasks may not be scheduled.

F. Resource-Aware-Scheduling algorithm

RASA is an innovative scheduling method. It considers the distribution and scalability characteristics of resources present in the cloud. The algorithm is built through a detailed study and analysis of two well known task scheduling algorithms, Max-min and

©IJRASET 2015: All Rights are Reserved

Min-min. RASA uses the merits of the both algorithms and covers their demerits. In order to achieve this, RASA first estimates the completion time each of the tasks. And after applies Min-min and Max-min algorithms alternatively. RASA uses the Min - min strategy to execute the small tasks prior to the larger ones. It also applies the Max-min strategy to minimize delays in the execution of large tasks and to support concurrency in the execution of small and large tasks.

G. Heterogeneous Earliest Finish Time

Heterogeneous Earliest Finish Time (or HEFT) is a heuristic to schedule a set of dependent tasks onto a network of heterogeneous workers taking communication time into account. HEFT takes a set of tasks as input. These tasks are then represented as a directed acyclic graph which includes a set of workers, the time each task execute on each worker, and the time to pass the results from each job to each of its children. This scheduling method is especially intended for heterogeneous systems. This algorithm does not consider the monetary costs. The major goal of HETF is to reduce the workflow makespan. HEFT includes two phases. First phase includes prioritizing tasks and the second phase includes assigning tasks to workers. In the second phase each worker is assigned some task. All tasks are prioritized and each one is scheduled, starting with the highest priority. The task which is of the highest priority and for which all dependent tasks have finished is scheduled on the worker and this leads to the earliest finish time of that task. The finish time of the task depends on the communication time to send all necessary inputs to the worker, and the time when that processor becomes available. In hard situations it is not easy to find the optimal scheduling using HEFT. HEFT is essentially a greedy algorithm and incapable of making short-term sacrifices for long term benefits. The Heterogeneous Earliest Finish Time algorithm selects the task with the highest upward rank at each step, and then the task is assigned to the most suitable processor that minimizes the earliest finish time [4].

III. PROPOSED SYSTEM

Heuristics uses mainly three key operators—transition, evaluation, and determination. The main intention is to search for the possible solutions on the convergence process. The transition operator creates the solution s. The evaluation operator measures the fitness of s by using a predefined measurement. And after that the determination operator determines the next search directions based on the s from the transition operator and the evaluation operator.

This heuristic scheduling mainly involves the following steps. First the parameters are set up, such as the maximum size of the population and the maximum number of iterations. Then the tasks that are needed to be scheduled are given as input to this method. Then initialize the initial population of solutions. Select a heuristic algorithm from the candidate pool of low level heuristics. Until the termination condition is not met, update the population of solutions from the candidate pool. And based on the improvement detected, randomly select a new heuristic. Output the best solution obtained so far. Genetic and Simulated annealing algorithms are included in the low level candidate pool. Genetic algorithm is a mainly a search heuristic. It mimics the natural selection process. This heuristic is used frequently to generate useful solutions to both search problems and optimization problems. It belongs to the class of evolutionary algorithms, and generate solutions to problems using techniques such as inheritance, selection, crossover and mutation. It combines the exploitation of best solutions from past searches with the exploration of new regions of the solution space. Any solution is represented by a chromosome. The quality of chromosome is determined by the fitness function and its value indicates how good the individual is compared to others in the population. Here evolution usually starts from a population of randomly generated individuals. This process is an iterative process. The population in each iteration called a generation. The more fit individuals are selected from the current population based on the fitness function. After that crossover and mutation is done. This new generation is used in the next iteration. The algorithm terminates when either a maximum number of generations has been produced. The output of this genetic algorithm is then fed into other evolutionary algorithms such as differential evolution.

Simulated annealing (SA) is a heuristic method mainly used in a large search space. It is used mainly when the search space is discrete. The inspiration comes from annealing in metallurgy. It is a technique of heating and cooling of a material to increase the size of its crystals. The same amount of cooling brings the same amount of decrease in temperature and it will bring a bigger or smaller decrease in the thermodynamic free energy. This slow cooling is implemented in the Simulated Annealing algorithm[5]. In each step, the SA heuristic considers some neighbouring state of the current state *c*, and decides moving the system to the new state or staying in state c. After doing these repeatedly the system moves to the states of lower energy. This iteration is done until a given computation budget has been exhausted or the system reaches a state that is good enough for the application. The probability of making the transition from the current state to a candidate new state is specified by an acceptance probability function. It depends on the energies of the two states, and on a global parameter, that is the temperature. Always states with smaller energies are better than those with a greater energy. Initially the temperature is set to a very high value. And in each step of the iteration it is decreased based on some annealing schedule. The probability that the SA algorithm moves from the current state to the neighbouring state is specified by transition probability function. This probability depends on the current

©IJRASET 2015: All Rights are Reserved

temperature.

IV. CONCLUSION

Rule-based scheduling algorithms have been widely used on many cloud computing systems because of their simplicity and easiness to implement. By using heuristic algorithms, the scheduling performance can be further increased. After combining genetic with differential evolution, it is seen that its performance has been increased. It is observed that the combined heuristic algorithm using genetic and simulated annealing produce better results than when each of this is used individually. Maximum tardiness, maximum lateness, maximum flowtime and makespan are used as the performance metrics.

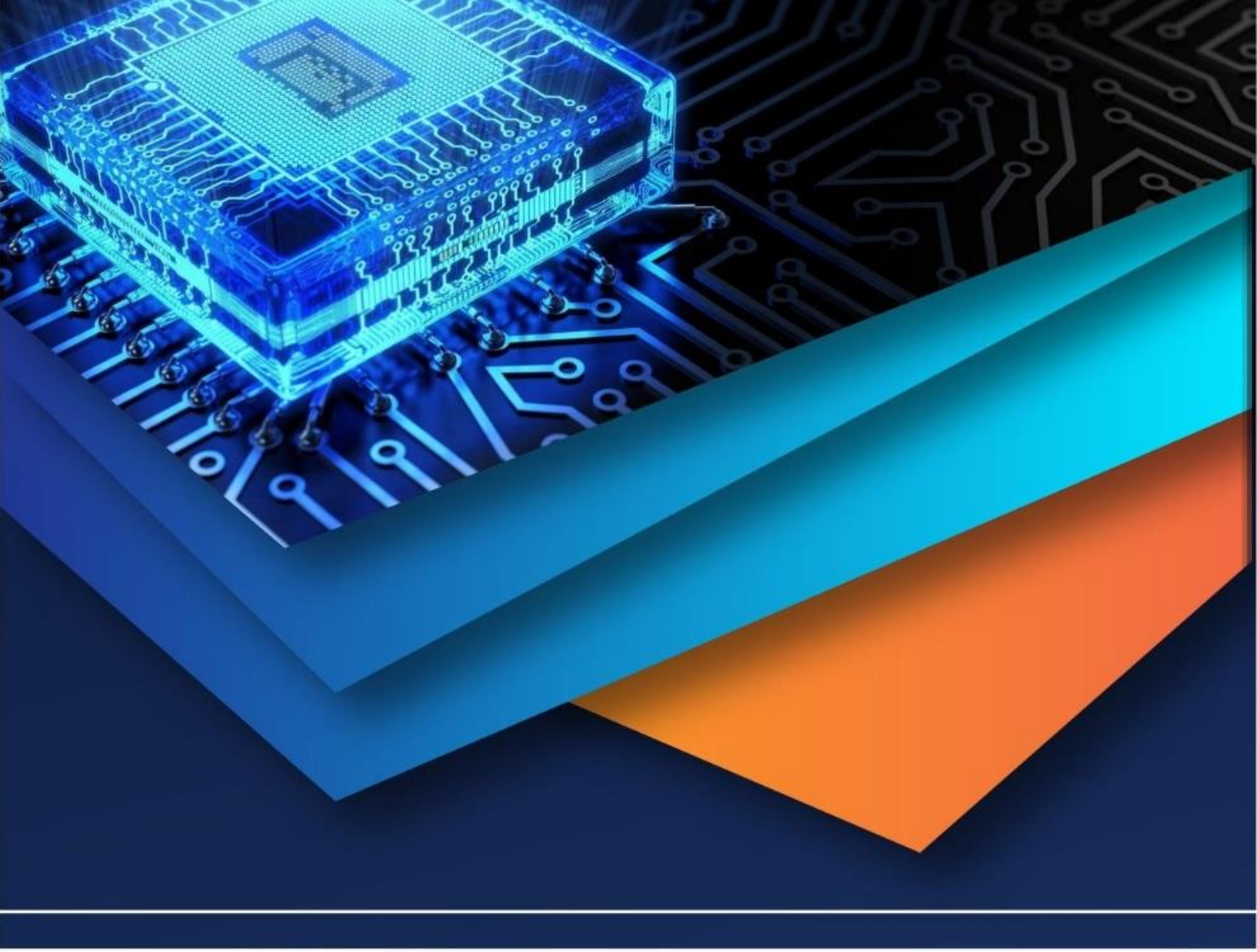
V. ACKNOWLEDGEMENT

The authors would like to thank professors of NSSCE, Palakkad for suggestions and support on this paper.

REFERENCES

[1] Chun-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang, and Chu-Sing Yang, "Hyper heuristic scheduling algorithm for cloud", IEEE TRANSACTIONS ON CLOUD COMPUTING, vol. 2, no. 2, april-june 2014.
[2] Yossi Azar, Navendu Jain, Nikhil R. Devanur, Naama Ben Aroya, "Cloud Scheduling with Setup Cost", Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures, pp.298-304, 2013.
[3] N. Kaur, T. S. Aulakh, and R. S. Cheema, "Comparison of workflow scheduling algorithms in cloud computing," Int. J. Adv. Comput.Sci. Appl., vol. 2, no. 10, pp. 81–86, 2011.
[4] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, "Task Scheduling Strategies for Workflow-based Applications in Grids", 05 Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid , vol.2, pp.759-767, 2005.
[5] S. Benedict and V. Vasudevan, "Scheduling of scientific workflows using simulated annealing algorithm for computational grids," Int. J. Soft. Comput., vol. 2, no. 5, pp. 606–611, 2007.
[6] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," in Proc. Workflows Support Large-Scale Sci., 2006, pp. 1–10.
[7] T. Chen, B. Zhang, X. Hao, and Y. Dai, "Task scheduling in grid based on particle swarm optimization," in Proc. Int. Symp. Parallel Distrib. Comput., 2006, pp. 238–245.
[8] W. N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," IEEE Trans. Syst., ssMan, Cybern., Part C: Appl. Rev.,vol. 39, no. 1, pp. 29–43, Jan. 2009.
[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw.: Practice Experience, vol. 41, no. 1,pp. 23–50, 2011.

©IJRASET 2015: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)