IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 2, Issue 4 (July-Aug. 2012), PP 15-16
www.iosrjournals.org

Resource Scheduling In Cloud Using Bee Algorithm For Heterogeneous Environment

Pradeep.R[1], Kavinya.R[2]
[1,2](Computer Science, Anna University, India)

Abstract : *The concept of cloud computing is gaining large exposure now, due to which, every industry aims at leasing resources and complete their jobs by the services provided by the cloud computing. The important service it provides is the Infrastructure as Service (IaaS) which helps company to lease computer resources in terms of memory or processing capacity. The major factor which determines the efficiency of resource utilization lies in scheduling of resources present at that point of time in a cloud. In this paper we will show how to effectively allocate the resource using one of the nature inspired algorithm, Bee algorithm.*
Keywords: *Cloud Computing; Resource Scheduling; Bee algorithm; Waggle; Scout job*

I. Introduction

The future of mankind lies in the power of cloud computing. Cloud computing satisfies the need of present era humans which is mobility. It renders the user his data how much big or small it is, the data is always kept within the reach of the user. The workload in cloud is heterogeneous because some work is CPU intensive or I/O intensive or other may require high processing ends [1]. It is also likely that the computing environment also is heterogeneous. The cloud has traditional servers in which some can be suitable for high end processing, some can be suitable for storage and some can be useful for I/O operations [2]. Hence a fitness function must be defined which matches the submitted job to its appropriate servers so that data center utilization is maximized. Also if every job is evaluated with a fitness function it would be time consuming. Hence time must be reduced along with a fitness function which points to job affinity [1]. It is proposed to use a waggle function to reduce this time consumption problem however the time for a job to reach its allocated node still persists. This paper is organized as follows. First, Bee algorithm is described followed by the proposed scheduling algorithm which takes care of heterogeneity and allocates resource efficiently is explained in Section 2 and in Section 3. Finally, the findings, conclusion and the future works are reported in Section 4.

II. Bee Algorithm

It is a nature inspired algorithm which tries to track the activities of bee to get their food. First they select scout bee to go and search a wide domain of areas, if a scout bee finds a potential food resource it returns to its hive and does a waggle dance which tells other bees the direction and the distance of the potential food resource. A set of selected bees goes to the food resource and starts bringing in the honey while other scout bee's does the same work and sets of bees are sent to different location to bring in the food. After every identification of a food resource the scout bee informs others and sets its course for other new sites nearby the potential food resource. Using this activities we define terms as in no of scout bees (n), no of sites selected out of n visited sites (m), no of best sites out of whole set (e), no of bees recruited for the best sites e (nep), no of bees recruited for other sites (m-e),

Bee's algorithm is as stated below:

1. Initialize population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met)
4. Select sites for neighborhood search.
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitness.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitness.
8. End While.

In first step, the bee's algorithm starts with the scout bees (n) being placed randomly in the search space. In step 2, the fitness of the sites visited by the scout bees are evaluated. In step 4, bees that have the highest fitness are chosen as "selected bees" and sites visited by them are chosen for neighborhood search. Then, in steps 5 and 6, the algorithm conducts searches in the neighborhood of the selected sites, assigning more bees to search near to the best e sites. The bees can be chosen directly according to the fitness associated with the sites they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighborhood of the best e sites which represent more promising solutions are made more detailed by recruiting more bees to follow them than the other selected bees. Together with scouting, this differential recruitment is a key operation of the Bees Algorithm. However, in step 6, for each patch only the bee with the

highest fitness will be selected to form the next bee population. In nature, there is no such a restriction. This restriction is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to its new population – those that were the fittest representatives from a patch and those that have been sent out randomly.

III. Scheduling Based on Bee Algorithm

In this section, the proposed algorithm for resource allocation using the above explained bee algorithm is justified. In this, meta-scheduler[3] is used, which sends independent jobs to various clusters present. In the beginning, the jobs will be submitted to meta-scheduler.

1.1 Select

The meta-scheduler using a select function will find a job which has lowest memory requirement, lowest I/O requirement and lowest processor required to complete their job which will act as a scout bees to find a site.

$$f(n) = \min\{U^{n}_{i=1}\ j(i)\} \quad (1)$$

where j(i) denotes the j^{th} job and function f(n) determines the minimum resource requirement job.

3.2 Fitness

The minimum resource requirement job which acts as a scout bee are identified and sent to the cluster where in the jobs identify the instances present [2]. A scout job identifies the site by using a fitness function which runs that job in a particular instance and if a progress is made it determines that instance specification as in it is either memory oriented or processor oriented.

Conceptually, fitness refers to how much progress each job is making with assigned resources compared to the same job running on the entire cluster. Therefore, it is between 0 (no progress at all) and 1 (progress is shown). The computing rate (CR) is used to calculate the fitness of an instance to the job. Specifically, given that CR(s; j) is the computing rate of slot s for job j, the progress share where Sj is a set of slots running tasks of job j and S is the set of all slots.

$$F(j) = CR(s;j) / CR(s';j) \quad (2)$$

3.3 Waggle

By identifying the sites resources the scouts returns to meta-scheduler and does the waggle function. Waggle function segregates the jobs present in meta-scheduler based upon scouts information like whether the instance is memory or processor oriented also its distance and cost to travel. The grouping takes place in such a way that the memory scout job passes on the information to the memory oriented jobs present and the selected set goes to the cluster to get executed. Care should be taken to select all jobs within instances capacity. If a job outruns instance capacity then the job has to wait until the scout jobs find another resource available adjacent to the just found site.

$$W(n)= \{\bigcup^{m}_{i=1}\ s(j) \in s\} \quad (3)$$

where the s job's resources which is an integral multiple of scout job i.e. s(j)

After waggle function the subsets of jobs are rendered to the desired sites by meta-scheduler and scout jobs sets course for another site exploration. Hence after the fitness function have to run only for the scout jobs in algorithm and the time is reduced drastically. Thus the resources are efficiently allocated also the time is reduced.

IV. Conclusion

In PSO algorithm pbest() and gbest() have to be calculated every time of the iteration in order to find the best job for the best resource[4]. Mesos [5] takes this approach to provide resource sharing and isolation across distributed applications. It adopts Dominent Resource Fairness(DRF) [6] as an example of resource allocation policy and leaves application level scheduling to each application. In this algorithm resource allocation is done but also every time it is not necessary to calculate the fitness function, if the pool of jobs utilise just the resources of the selected site then it can be easily allocated and fitness function need not be computed the second time. Many systems running in the cloud involve multilevel scheduling - resource allocation at infrastructure level and job scheduling at application level. It is planned to experiment with PS in a framework to be constructed on top of Mesos.

References

[1] Gunho Leey, Byung-Gon Chunz, Randy H. Katz, "Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud", University of California, Berkeley, Yahoo! Research

[2] Amazon ec2. http://aws.amazon.com/ec2.

[3] Marco A. S. Netto and Rajkumar Buyya, "Offer-based Scheduling of Deadline-Constrained Bag-of-Tasks Applications for Utility Computing Systems", IEEE International Symposium on Parallel&Distributed Processing, 2009.

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

[5] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos, "A platform for fine-grained resource sharing in the data center", ACM Workshop on Scientific Cloud Computing, 2011.

[6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types", in Proc. Proceedings of the 8th USENIX conference on Networked systems design and implementation, 2011.