IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Efficient Decentralized Job Tracking and Scheduling Using EBLA Algorithm

A Priyadarshini[1], Vaishnavi[2], V.Anbarasu[3]
[1, 2, 3]**Department of Information Technology, Jeppiaar Engineering College, Chennai, India**

Abstract

Cloud computing is known as digital service delivery over the internet by several applications which are carried out by computer systems in distributed datacenters. Due to novelty of cloud computing field, there exists many standard task scheduling algorithm used in cloud environment. In cloud as there is a high communication cost the resources are to be scheduled for optimum utilization. Researchers has involved in proposing job scheduling algorithms that are compatible and applicable in cloud computing environment as the users pay for resources. This paper proposes a distributed job tracking algorithm which communicates to job/meta broker that collects the information from the independent job trackers and informs job monitor. It supplies a high performance computing based on protocols which allow shared computation and storage over long distances. This paper proposes a decentralized method of the job tracker for status of jobs implements scheduling algorithm for jobs called Enhanced Bees Life Algorithm (EBLA) applied to efficiently schedule computation jobs among processing resources onto the cloud datacenters. The EBLA algorithm aims at spreading the workloads among the processing resources in an optimal fashion to reduce the total execution time of jobs and then, to improve the effectiveness of the whole cloud computing services. EBLA has been inspired by bees' life. Hence the proposed architecture involved with novel algorithm plays a vital role to get maximum benefit from the resources.
Keywords: *cloud computing, meta broker, Enhanced bees life algorithm, decentralized job tracker.*

1. Introduction

Cloud computing is a computing concept shared by a large number of computers through communication network (e.g. internet). Here the service is provided by the virtual hardware invoked by software running on those computers and it can also be passed around (like a cloud).

It provides the services through the enormous amount of resources available in a large number of nodes/ virtual computers connected through internet. Without the internet it is not possible for computing or accessing those services so it is also called as Internet Computing. It is a distributed system where there are collection of computing services provided and communication resources located in distributed datacenters which are shared by several end users. There are four kinds of the cloud; they are public cloud, private cloud, hybrid cloud, and finally community cloud. A public cloud is the standard cloud computing one in which service provider makes resources such as storage and application over the internet. It may be free of cost and offered on a pay-per-usage mode

The advantages of the cloud are:

- It is easy and cost inexpensive because hardware, application used and bandwidth cost are low and are easily provided by the provider.
- User whatever used resources should be payable one so doesn't have wasted resources.
- It is very easy to meet user's needs.

Cloud provides all the below services such as storage, database, information, process, application, platform, integration, security, management/ governance, testing, infrastructure.

Cloud is almost preferred than any similar type as it provides more security when compared to others. Though the communication cost is higher, it has a wider scope and so many researchers are proposing their models for ensuring security for cloud and optimal scheduling of the jobs submitted by the user. Scheduling is the most important part of the cloud computing wherein the workloads is distributed among thousands of datacenters for solving the queries of tens of thousands users. In such cases the resources available should be effectively managed so that the response time for allocating jobs also gets reduced .This paper also has the aim of increasing

the computational speed by effectively utilizing the resources by providing a decentralized job tracker for collecting the status of the jobs running over the datacenters and then reporting to the job scheduler wherein the job scheduler schedules the job using Enhanced Bees life Algorithm which is an updated version of bees life Algorithm.

2. Related Work

A. State of the art

In the past years, few research activities has been done successfully in the cloud computing environment.

Amazon Elastic Compute Cloud (EC2) is an IaaS cloud, and is the most well known cloud. Through the use of virtual machines EC2, clients create as many (virtual) machines they require and then host all their required services after an optimized scheduling within the machines. EC2 clients are charged per CPU hour per virtual machine. It is even possible to create a cluster within EC2 by requesting EC2 to create multiple instances of a base virtual machine [1].

The load balancing framework for high-performance clustered storage systems provides reconfiguring a system facing workload changes. It simultaneously balance load and reduces the cost. The main purpose is to provide automatic reconfiguration or to present an administrator with a range of near optimal reconfiguration, allowing a tradeoff between load distribution and reconfiguration cost. It simply balances the workload on a NetAppData ONTAP GXsystem, a commercial scale-out clustered NFS server implementation [2].

pMapper-Power and migration cost aware application placement in virtualized systems. Here in this workload placement on servers has been traditionally driven by objects. Design, implementation and evaluation of a power-aware application placement controller in the context of an environment with heterogeneous virtualized server clusters to be investigated. It takes the power consideration and cost efficient. The contribution is of two-fold: to capture cost-aware application placement problem may need to various settings. In second, the pMapper architecture and placement algorithms to solve one practical formulation of problem: reduce the power to a fixed requirement of performance [3].

The fourth paper present a new Bee Swarm optimization algorithm called Bees Life Algorithm (BLA) applied to efficiently schedule computation jobs among processing resources onto the cloud datacenters. It is considered as NP-Complete problem and it aims at spreading the workloads among the processing resources in an optimal fashion to reduce the total execution time of jobs and then, to improve the effectiveness of the whole cloud computing services. BLA has been inspired by bees' life in nature represented in their most important behaviours which are reproduction and food source searching [4].

B. Problem statements and notations

Job Scheduling aims at allocating the jobs to datacenters that are available in the cloud which have free memory space to execute the task so that the execution time (job span) of the overall tasks of jobs is minimized. This problem can be formulated as follows. We denote it by
Jobs = {J1, J2... Ji… Jn} has *'n'* jobs to be scheduled.
Here instead of directly communicating the datacenters for the status of jobs running in it and the job monitor is first contacted and then the meta-broker is contacted which will always be in contact the job trackers which has one-to-one relationship with the datacenters (ie.., decentralized)so it will be always ready to provide the status of the jobs running. The meta-broker will provide the status to the job to the job scheduler. With the aid of the information provided by the meta-broker the job scheduler will schedules the job(by even segmenting the jobs) by following bees algorithm in the datacenters which is having the resources accordingly which best fits for the job segment to be executed.

First if there are n datacenters then the job trackers available are
JobTrackers={JTrack1,JTrack2,....,JTrackn}**.....(1)**
and they return the available resources at each datacenters. Moreover, as we said , we consider for each job *'i'*: there exist a number of partitions as below
JobiTasks = {JTaski1, JTaski2,...,JTaskim}**.....(2)**
here, *'m'* task partitions of Jobi disseminated/distributed among *'m'* cloud datacenters *(DCs)* which is available and suitable in order to be executed. Consequently, each cloud datacenter can carry out a disjoint subset of the

decomposed/divided jobs set. For datacenter's assigned jobs, *DCj* ensures the execution of their tasks as follows

DCjTasks = {JTaskaj, JTaskbj,..., JTaskrj} .....(3)

where ***0<r<n***

The union of these overall disjoint subsets gives the whole set of jobs. For example, if the jth datacenter *DCj* carries out

DCjTasks = {JTask1j, JTask2j,..., JTask9j} .....(4)

which are tasks of jobs *J1, J2 …,* and *J9* completely along with the partitions of job respectively.

Therefore, the total execution time of all job tasks (*'r'* tasks) assigned to DCj would be

Jobspan(DCjTasks) =Max (JTaskkj.StartTime + JTaskkj.ExecTime) .....(5)

job task *'k' JTaskkj* starts executing on *DCj* and *JTaskkj.ExecTime* is the execution time of *JTaskkj* at *DCj*.

Thus, the job scheduling problem in the cloud computing could be defined as searching of a set of datacenters which are free that best fits the job as follows:

DCTasks = {DC1Tasks, DC2Tasks.., DCmTasks}.....(6)

which contains set of datacenters and: for any j datacenter with resources,

DCjTasks = {JTaskaj, JTaskbj, ... JTaskrj} .....(7)

where $0 < r \leq n$ assigned with tasks from different jobs such that the job span gets reduced.

In order to evaluate the quality of the requested solution *(DCTasks)*, a bestfit function is defined as follows (used to calculate the above *jobspan*):

Bestfit(DCTasks) = Σ (Bestfit(JTaskij, DCj)).....(8)

where $(1 \leq j \leq m)$ and

BestFit(JTaskij,DCj)= JTaskij.ExecTime

where, *JTaskij.ExecTime* is the execution time of task of job *'i'* needs to run in *DCj. Bestfit* is the function which provides the job the datacenter which has the exact resource such that the time is minimized

3. Decentralized Job Tracking and Enhanced Bees Life Algorithm

3.1 Decentralized Job Tracking

The decentralized job tracking is that the job tracker who holds the information about the jobs running in the datacenters is associated with it using one-to-one relationship. So that the concentration of the job tracker is more such that the speed of the whole system get increased. The metabroker which is presented will have the information about all the job trackers and get through the information from them to the job monitor and then to the job scheduler who will allocate the jobs submitted to them by the user. While allocating each jobs they are divided into small executable parts and executed in different datacenters based on the type of resource available and the amount of resource available etc..,

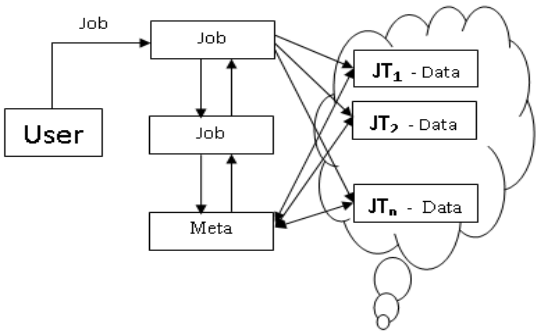

Figure 1. Architecture for Job Scheduling in Cloud

3.2 Bees in nature

The bees feed on nectar as a source of energy in their lives and use pollen as a source of protein in the rearing larvae. It is native to Europe and Africa .The population of bee actually consist of three partitions namely the queen, worker and drone. Here the queen is actually the fittest person and next the workers and next the drone. The queen will mate to the entire drone and the larvae which has the enormous proteins and energy/food will become the queen for next time.

3.3 Bees Life Algorithm Description

Let there is 1 queen (only one queen is possible), D drones and W workers of which the fittest is the queen. The two characters which characterize the bees are the searching of food and the reproduction/ the mating using the crossover and the mutation concepts. There we have to find D fittest broods for drones and W fittest broods for the workers. If there exist brood which is better than the queen then the current brood is assumed as the next queen.

3.4 Relating to the problem

There exists D jobs such that the exact place for have to found out from the W datacenters and the fittest place for each job is found by checking with each place which are free that is already found out by the job tracker. For each job its queen place is found out in this manner.

3.5 Algorithm

1. Find out the datacenters with exact resources present in it with the help of decentralized job trackers.

2. Report it to the metabroker the details of the datacenters with the information of resources in it.

3. Report the details to the job monitor and in turn to the job scheduler who schedules the job submitted to them.

4. There are about N jobs submitted to the job scheduler by the users.

5. And Let us assume that there are N datacenters with certain amount of the free space in it which is specified by the job tracker to the job scheduler.

6. The jobs which are submitted by the users are divided into small segments based upon the characteristics and then each part is assigned to the best place.

7. The best place is found out by passing through the datacenters as the bees pass through the broods by using the crossover and mutation methods which is discussed later in this same algorithm.

8. Start the loop where for each part of work the suitable datacenter is found and for each jobs it is done.

9. We can see that a datacenter will execute the tasks (segmented jobs) from different jobs and so we have to retrieve the information properly and the combined to give the result for the users.

10. Apply the best fit function at each case as that at each case the fittest for a particular part of the job will vary.

11. Thus the total time taken is found at last as Jobspan (DCjTasks) =Max (JTaskkj.StartTime + JTaskkj.ExecTime) as the start time and the execution time for each task in each of the datacenter.

12. Continue the loop until the each task of each job is finished.

13.End loop.

3.6 Crossover

It is the process of queen mating with randomly chosen drone with the probability. They mate with each other to form the children which are in the range of space. It can be represented as

Before crossover: Parents
Pi={<JTaskai, JTaskbi, JTaskci, JTaskdi, JTaskei,...,JTaskri>}
Pj={<JTaskAj, JTaskBj, JTaskCj, JTaskDj, JTaskEj,...,JTaskRj>}

After crossover: Children
Pi={<JTaskai, JTaskBi, JTaskCi, JTaskDi, JTaskei,...,JTaskri>}
Pj={<JTaskAj, JTaskbj, JTaskcj, JTaskdj, JTaskEj,...,JTaskRj>}

3.7 Mutation

It is a process by which a random selected process is replaced by another process. It is done with the probability Pm. It can be represented as

Child before mutation:

Ci={<JTaskai, JTaskbi, JTaskci, JTaskdi, JTaskei,...,JTaskri>}

Child after mutation:
Ci={<JTaskai, JTaskXi, JTaskci, JTaskdi, JTaskei,...,JTaskri>}

3.8 Greedy algorithm

It is the nearest datacenter that suits well for the job and also it is nearby. It is only local and can be represented as follows.

Original individual:
DCiTasks={<JTaskai,JTaskbi,JTaskci, JTaskdi,...,JTaskri>}
We assume that *DCjTasks* is the nearest datacenter to *DCiTasks*, where:
DCjTasks={<JTaska'j,JTaskYj,JTaskc'j, JTaskd'j,...,JTaskr'j>}

The neighbour individual will be:
DCiTasks={<JTaskai,JTaskYi,JTaskci, JTaskdi,...,JTaskri>}

3.9 BLA Complexity algorithm

First the time taken to allocate each jobs of N can be represented as T (N). The time taken by the job to be allocated can be represented as

T(NS×(N+((N×pc)×pm)+(FBest×B)+(FOther×(W-B)))

The process can be executed in the best case *NS* times. During the iteration, the generated number of off springs is *(N × pc)* with *pc*- crossover probability. Among them, *((N × pc) × pm)* will be mutated where *pm* is the mutation probability. Next, *W* workers among *N* ensure the food source foraging as a greedy local search. FBest foragers are sent to B best regions and FOthers foragers are sent to the remainder regions (W-B) regions. Therefore, FBest perform O(FBest × B) time units however, FOther execute O(FOthers × (W-B)) time units.

4. Implementation

First let us associate the job trackers for each datacenters and perform all the tracking process and calculate TS and execute to calculate TE. Finally calculate the TT to compare.

We are considering 4 jobs which are going to be executed let us compare the execution of the jobs using the BLA and EBLA algorithms

Dc	1	2	3	4	5	6	7	8	9	10
Exec Time in ms	10	10	10	12	10	12	10	10	10	12

Figure 2. Execution time of the partitions of job

Job1Tasks={JTask1 1, JTask1 4, JTask1 6, JTask1 7, JTask1 8}
Job2Tasks={JTask2 1, JTask2 5, JTask2 7, JTask2 8, JTask2 9}
Job3Tasks={JTask3 1, JTask3 6, JTask3 7, JTask3 9, JTask3 3}
Job4Tasks={JTask4 1, JTask4 4, JTask4 7, JTask4 8, JTask4 10}

Total time taken to execute the job completely (TT) = Time taken for tracking the status of jobs running in datacenter (TS) + Time taken for execution (including allocating of jobs) of jobs(TE).

In BLA, TE= (10+12+12+10+10) + (10+10+10+10+10) + (10+12+10+10+10) + (10+12+10+10+12) = 54 + 50 + 52 + 54 =210 ms
Assume say ,TS =100 ms

TT= 210+100 = 310 ms

In EBLA, similar to BLA the TE is TE= (10+12+12+10+10) + (10+10+10+10+10) + (10+12+10+10+10) + (10+12+10+10+12) = 54 + 50 + 52 + 54 = 210 ms, TS= 50 ms

TT= 210+ 50 = 260.

EBLA by consumes less amount of time 260/310*100 = 83.87% of time of BLA.

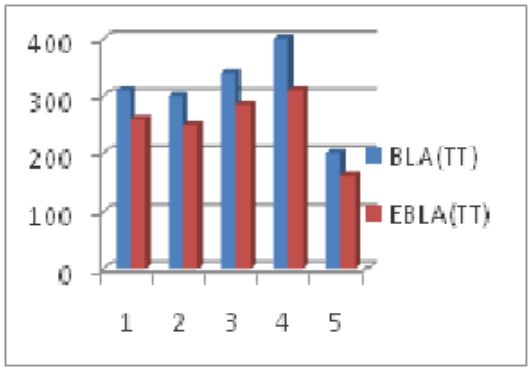

Figure 3: Graphical representation of time consumption of BLA and EBLA

The representing BLA and EBLA are as follows:

While computing them using the concept of speed the EBLA is about 40% efficient while comparing the speeds as it has a high speed. It can be represented as follows

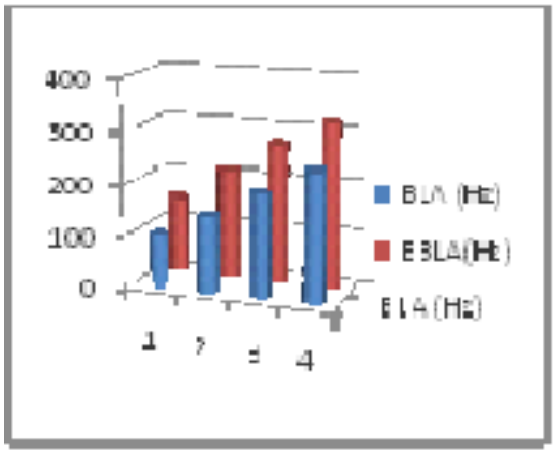

Figure 4. Comparison of speed of EBLA and BLA

5. Conclusion

In this paper we discussed the decentralized job tracker to track the information of the data centers which are running and the status of jobs running and the resources available such as CPU memory remaining etc.., we also discussed the role of the job monitor present. We discussed the bee's algorithm for scheduling the jobs to datacenters which are free with the best fit concept used. The experimentation of this concept provided a greater result as the speed is enhanced by the decentralized job trackers. The time consumed for the allocation of jobs is also proved in above case with the graph designed for both. The results provided the efficiency and performance of the proposed algorithm. It thus serves as a real time scheduler. And so it is very much useful for application and execution of real time jobs.

References

[1] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/ (accessed on October 25, 2011)

[2] D. Kunkle, and J. Schindler, "A Load Balancing Framework for Clustered Storage," LNCS, vol. 5374, pp. 57–72. Springer, 2008.

[3] A. Verma, P., Ahuja, A., and Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," LNCS, vol. 5346, pp. 243–264, Springer, 2008.

[4] Salim Bitam, "Bees life Algorithm for job scheduling in cloud computing" The Genetic and Evolutionary Computation Conference GECCO2001, San Francisco, USA, 2001.

Authors

First author – **A.Priyadarshini** is currently doing her Third year **B. TECH Information Technology** at **Jeppiaar Engineering College** in Chennai, Tamil Nadu .She has presented a paper based on cloud computing technical symposium and has attended many workshops.
Email – darshinianbalagan1994@gmail.com.

Second author – **R.Vaishnavi** is currently doing her Third year B. TECH Information Technology at Jeppiaar Engineering College in Chennai, Tamil Nadu .She has presented a paper based on cloud computing in technical symposium and has attended many workshops.
Email – vaishnavi.theba@gmail.com

Third author – **V. Anbarasu B.E., M.Tech., (Ph.D)** is working as an **Associate Professor** in the **Department of Information Technology at Jeppiaar Engineering College** in Chennai, Tamilnadu. His areas of interest are Operating Systems, Human Computer Interface and Programming Paradigm. He has 10 years of teaching experience. He has presented 12 papers in International and National Conferences and also published 5 papers in International and National journals. He has attended several workshops and FDPs.
Email ID - anbarasukv@gmail.com