RESEARCH ARTICLE OPEN ACCESS

Performance Evaluation of Meta-heuristic based Load Balancing in Cloud Computing

Pooja Mangla [1], Sandip Kr. Goyal [2]
PhD Scholar [1], PhD Guide [2] HOD
Department of Computer Science and Engineering
Maharish Markandeshwar University, Mullana
India

ABSTRACT
Load balancing is an optimization technique. In the simplest load-balancing mechanisms, the load balancer listens to a network port for service requests. When a request from a client or service requester arrives, the load balancer uses a scheduling algorithm to assign where the request is sent. It can be used to increase utilization and throughput, lower latency, reduce response time, and avoid system overload.In this paper, we are going to implement GA and PSO meta-heuristic techniques and we will evaluate makespan for both of these algorithms.
Keywords:- Cloud Computing, Load Balancing, Genetic Algorithm, Particle Swarm Optimization.

I. INTRODUCTION

In a dynamic economic environment, your company's survival may depend on your ability to focus on core business and adapt quickly. Cloud Computing is everywhere. Cloud computing is completely internet-based technology which incorporates the concept of parallel and distributed computing to provide shared resources, which includes hardware, software and information to computers or other devices on demand. Cloud computing works on pay-as-per-usage technique in which user can use as many resources for which he has paid.

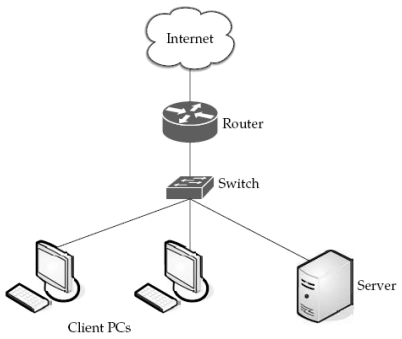

Fig. 1 A cloud is used in network diagrams to depict the Internet

The cloud icon represents "all that other stuff" that makes the network work. It's kind of like "etc." for the rest of the solution map[1]. It also typically means an area of the diagram or solution that is someone else's concern. Cloud computing promises to cut operational and capital costs and, more importantly, let IT departments focus on strategic projects instead of keeping the datacenter running.

In this paper, we look at cloud computing from three perspectives: the strategy from both the customer and the provider's point of view, business and economic considerations, and the technical underpinnings. We also examine how companies are using the cloud to control IT expenditures as they prepare to move to a service-centric world[3].

Many players make up the world of cloud computing:

✓ The vendors providing applications and enabling technology, infrastructure, hardware, and integration

✓ The partners of these vendors that are creating cloud services offerings and providing support services to customers

✓ The business leaders themselves who are either using or evaluating various types of cloud computing offerings

The cloud in cloud computing provides the means through which everything — from computing power to computing infrastructure, applications, business processes to personal collaboration — can be delivered to you as a service wherever and whenever you need. Cloud computing is offered in different forms[3]:

✓ Public clouds

✓ Private clouds

✓ Hybrid clouds, which combine both public and private

In general the cloud — similar to its namesake of the cumulus type — is fluid and can easily expand and contract. This elasticity means that users can request additional resources on demand and just as easily deprovision (or release) those resources when they're no longer needed. This elasticity is one of the main reasons individual, business and IT users are moving to the cloud

II. LOAD BALANCING

One characteristic of cloud computing is virtualized network access to a service. No matter where you access the service, you are directed to the available resources. The technology used to distribute service requests to resources is referred to as *load balancing*. Load balancing can be implemented in hardware, as is the case with F5's BigIP servers, or in software, such as the Apache mod_proxy_balancer extension, the Pound load balancer and reverse proxy software, and the Squid proxy and cache daemon.

The objective of a load balancing is to distribute the incoming requests onto a set of physical machines, each hosting a replica of an application, so that the load on the machines is equally distributed [4]. The load balancing algorithm executes on a physical machine that interfaces with the clients. This physical machine, also called the front-end node, receives the incoming requests and distributes these requests to different physical machines for further execution.

This set of physical machines is responsible for serving the incoming requests and are known as the back-end nodes. Typically, the algorithm executing on the front-end node is agnostic to the nature of the request. This means that the front-end node is neither aware of the type of client from which the request originates nor aware of the category (e.g., browsing, selling, payment, etc.) to which the request belongs to. This category of load balancing algorithms is known as class-agnostic. There is a second category of load balancing algorithms that is known as class-aware. With class-aware load balancing and requests distribution, the front-end node must additionally inspect the type of client making the request and/or the type of service requested before deciding which back-end node should service the request. Inspecting a request to find out the class or category of a request is difficult because the client must first establish a connection with a node (front-end node) that is not responsible for servicing the request. Figure 16.5 shows the general taxonomy of different load balancing algorithms.

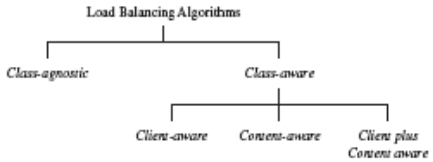

FIGURE 16.5. General taxonomy of load-balancing algorithms.

Fig. 2 General taxonomy of load-balancing algorithms

A session ticket is created by the load balancer so that subsequent related traffic from the client that is part of that session can be properly routed to the same resource. Without this session record or persistence, a load balancer would not be able to correctly failover a request from one resource to another. Persistence can be enforced using session data stored in a database and replicated across multiple load balancers. Other methods can use the client's browser to store a client-side cookie or through the use of a rewrite engine that modifies the URL. Of all these methods, a session cookie stored on the client has the least amount of overhead for a load balancer because it allows the load balancer an independent selection of resources.

The more sophisticated load balancers are workload managers. They determine the current utilization of the resources in their pool, the response time, the work queue length, connection latency and capacity, and other factors in order to assign tasks to each resource. Among the features you find in load balancers are polling resources for their health, the ability to bring standby servers online (priority activation), workload weighting based on a resource's capacity (asymmetric loading), HTTP traffic compression, TCP offload and buffering, security and authentication, and packet shaping using content filtering and priority queuing.

An Application Delivery Controller (ADC) is a combination load balancer and application server that is a server placed between a firewall or router and a server farm providing Web services. An Application Delivery Controller is assigned a virtual IP address (VIP) that it maps to a pool of servers based on application specific criteria. An ADC is a combination network and application layer device. You also may come across ADCs referred to as a content switch, multilayer switch, or Web switch.

In a cloud environment, executing application requests on underlying grid resources consists of two key steps. The first, which we call VM Provisioning, consists of creating VM instances to host each application request, matching the specific characteristics and requirements of the request. The second step is mapping and scheduling these requests onto distributed physical resources (Resource Provisioning). Most virtualized data centers currently provide a set of general-purpose VM classes with generic resource

configurations, which quickly become insufficient to support the highly varied and interleaved workloads. Furthermore, clients can easily under- or overestimate their needs because of a lack of understanding of application requirements due to application complexity and/or uncertainty, and this often results in over-provisioning due to a tendency to be conservative.

The decentralized clustering approach specifically addresses the distributed nature of enterprise grids and clouds. The approach builds on a decentralized messaging and data analysis infrastructure that provides monitoring and density-based clustering capabilities. By clustering workload requests across data center job queues, the characterization of different resource classes can be accomplished to provide autonomic VM provisioning. This approach has several advantages, including the capability of analyzing jobs across a dynamic set of distributed queues, the nondependency on a priori knowledge of the number of clustering classes, and the amenity for online application and timely adaptation to changing workloads and resources. Furthermore, the robust nature of the approach allows it to handle changes (joins/leaves) in the job queue servers as well as their failures while maximizing the quality and efficiency of the clustering.

Load balancing is an optimization technique; it can be used to increase utilization and throughput, lower latency, reduce response time, and avoid system overload. The following network resources can be load balanced:

- Network interfaces and services such as DNS, FTP, and HTTP
- Connections through intelligent switches
- Processing through computer systemassignment
- Storage resources
- Access to application instances

Without load balancing, cloud computing would very difficult to manage. Load balancing provides the necessary redundancy to make an intrinsically unreliable system reliable through managed redirection. Load balancing is early always a feature of server farms and computer clusters and for high availability applications.

A load-balancing system can use different mechanisms to assign service direction. In the simplest load-balancing mechanisms, the load balancer listens to a network port for service requests. When a request from a client or service requester arrives, the load balancer uses a scheduling algorithm to assign where the request is sent. Typical scheduling algorithms in use today are round robin and weighted round robin, fastest response time, least connections and weighted least connections, and custom assignments based on other factors.

III. META-HEURISTIC ALGORITHM

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

A. Genetic Algorithm

The most popular technique in evolutionary computation research has been the *genetic algorithm*. In the traditional genetic algorithm, the representation used is a *fixed-length bit string*. Each position in the string is assumed to represent a particular feature of an individual, and the value stored in that position represents how that feature is expressed in the solution. Usually, the string is "evaluated as a collection of *structural* features of a solution that have little or no interactions". The analogy may be drawn directly to genes in biological organisms[7]. Each gene represents an entity that is structurally independent of other genes. The main reproduction operator used is *bit-string crossover*, in which two strings are used as parents and new individuals are formed by swapping a sub-sequence between the two strings (see Fig. 3). Another popular operator is *bit-flipping mutation*, in which a single bit in the string is flipped to form a new offspring string

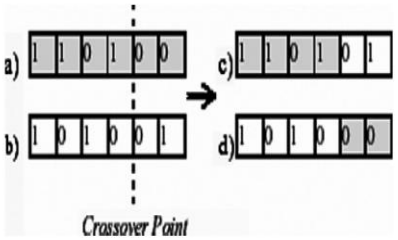

Fig. 3 Bit-string crossover of parents a & b to form offspring c & d

(see Fig. 4). A variety of other operators have also been developed, but are used less frequently (e.g., *inversion*, in which a subsequence in the bit string is reversed). A primary distinction that may be made between the various operators is whether or not they introduce any new information into the population. Crossover, for example, does not while mutation does. All operators are also constrained to manipulate the string in a manner consistent with the structural interpretation of genes. For example, two genes at the same location on two strings may be swapped between parents, but not combined based on their values. Traditionally, individuals are selected to be parents *probabilistically* based upon their fitness values, and the offspring that are created replace the parents. For example, if N parents are selected, then N offspring are generated which replace the parents in the next generation.

Fig. 4 Bit-flipping mutation of parent a to form offspring b

The Genetic algorithm process is discussed through the GA cycle in figure 5.

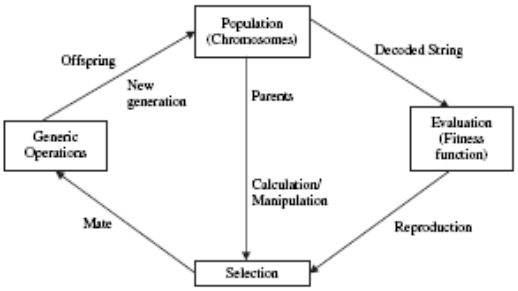

Fig. 5 Gentic Algorithm cycle

The flowchart showing the process of GA is as shown in Fig.6.

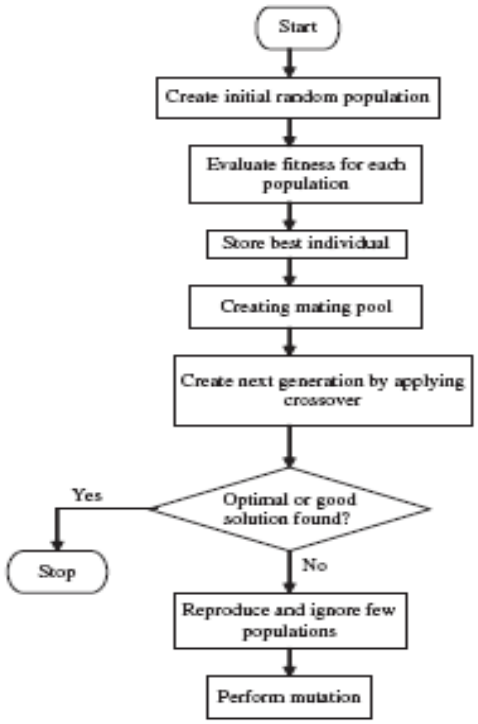

Fig. 6 Flowchart of Genetic Algorithm

Reproduction is the process by which the genetic material in two or more parent is combined to obtain one or more offspring. In fitness evaluation step, the individual's quality is assessed. Mutation is performed to one individual to produce a new version of it where some of the original genetic material has been randomly changed. Selection process helps to decide which individuals are to be used for reproduction and mutation in order to produce new search points.

Before implementing GA, it is important to understand few guidelines for designing a general search algorithm i.e. a global optimization algorithm based on the properties of the fitness landscape and the most common optimization method types:

1. determinism: A purely deterministic search may have an extremely high variance in solution quality because it may soon get stuck in worst case situations from which it is incapable to escape because of its determinism. This can be avoided, but it is a well-known fact that the observation of the worst-case situation is not guaranteed to be possible in general.
2. non-determinism: A stochastic search method usually does not suffer from the above potential worst case "wolf trap" phenomenon. It is therefore likely that a search method should be stochastic, but it may well contain a substantial portion of determinism, however. In principle it is enough to have as much non-determinism as to be able to avoid the worst-case wolf traps.
3. local determinism: A purely stochastic method is usually quite slow. It is therefore reasonable to do as much as possible efficient deterministic predictions of the most promising directions of (local) proceedings. This is called local hill climbing or greedy search according to the obvious strategies.

B. Particle Swarm Optimization

PSO refers to a relatively new family of algorithms that may be used to find optimal (or sub-optimal) solution to numerical and qualitative problems. PSO was first introduced by Russell E berhart and James Kennedy in 1995 and used for optimizing of continuous non-linear functions. It is also related to evolutionary computation and has ties to both GA and evolutionary programming.

The basic idea behind PSO is that it keeps track of global best and self best. PSO uses a population of particles that fly over the fitness landscape in search of an optimal solution. The particles are controlled by forces that encourage each particle to fly back both towards the best point sampled by it and towards the swarms best point while its momentum tries to keep it moving in its current direction. The basic PSO includes three steps:

i) Generating a particle's position and velocity
ii) Velocity update
iii) Position update

As we already know that each particle remembers its own best position (pBest) it has encountered by them while all particles know about the overall best position (gBest) which has been found by all particles of the Swarm. PSO explores global optimal solution through exploiting the particles memory and the swarm's memory PSO can become one of the most important Swarm Intelligence (SI) techniques because of its

properties of low constraints on the continuity of objective function and joint of search space. Moreover, its ability of adapting to dynamic environment makes it different and a special method of SI. When compared to GA, PSO has no evolution operators such as crossover and mutation. This is expected to move the swarm toward the best solutions [8].

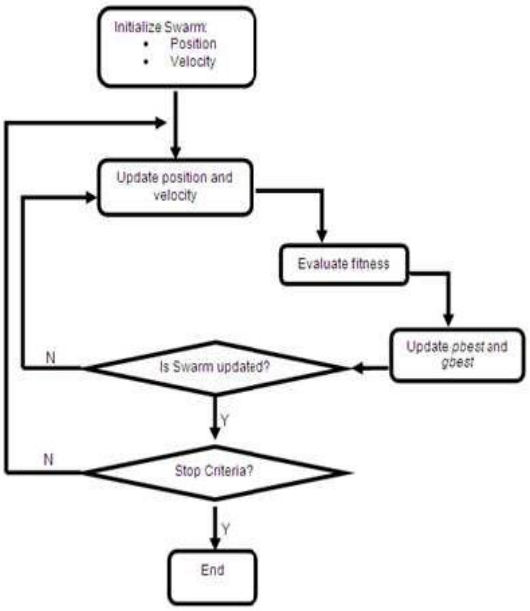

Fig. 7 Particle Swarm Optimization

Following is the template for PSO:

Random initialization of the whole swarm
Repeat
Evaluate f (xi)
For all particles i
Update velocities by Eq. (1)
Move to the new position by Eq. (2)
If f (xi) < f (pbi) Then pbi = xi
If f (xi) < f (gb) Then gb = xi
EndFor
Until stopping criteria
Output: Best solution found.

IV. RESULT AND DISCUSSIONS

In this paper, to analyze and compare the two meta-heuristic approaches-Genetic Algorithm and Particle Swarm Optimization, CloudSim tool is used. We have used various data centers and virtual machines. The performance can be evaluated on the basis of various parameters: scalability, performance, resource utilization, fault tolerance, associative overhead, but in our research, we have calculated Makespan for both of these algotihms, in which no. of tasks are 100 and Virtual Machines are 5. Also, makespan is evaluated based on no. of attempts. Following figures depict the results, generated with the help of CloudSim simulator.

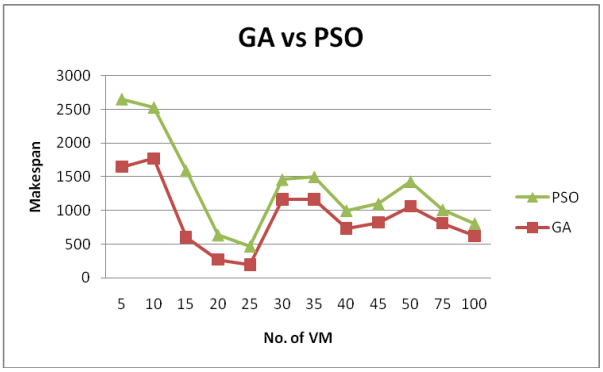

Fig. 7 Comparing Makespan of GA & PSO for different no. of VM

In above figure, the parameter makespan of GA and PSO are evaluated for fixed number of tasks, but different number of virtual machines. This result shows that in this scenario, PSO is showing better results.

The figure, given below, shows the comparison made on the basis of number of attempts.

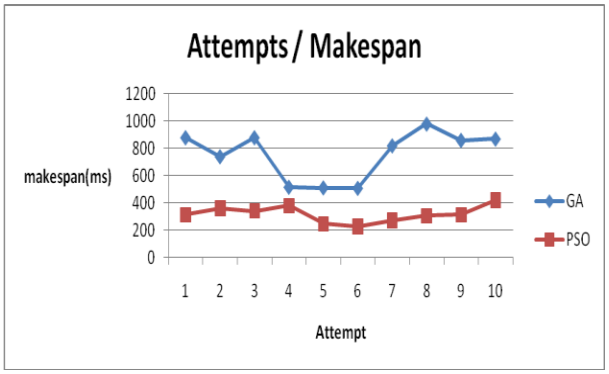

Fig. 8 Comparing Makes span of GA & PSO for various attempts

In above figure, again parameter makespan is evaluated on same number of tasks, i.e. equal to 100 and same number of Virtual Machines i.e., equal to 5. Here, Also, makespan of PSO is lesser than that of GA for this scenario which means, PSO is performing better than GA.

V. CONCLUSIONS

In this paper, we evaluated the makespan of GA and PSO based on number of tasks and number of virtual machines. The results are shown above, which depicts that PSO is performing better when the number of tasks are fixed and

number of virtual machines are increased. Also, results are evaluated for the number of attempts, in which tasks and virtual machines are fixed. With these results, we can show the random nature of

ACKNOWLEDGMENT

This research was supported by MMU, Mullana. Firstly, I would like to thank Dr. Sandip Kumar Goyal, who is guiding me in my research work and moderated this paper and in that line improved the manuscript significantly. Also, I am grateful to my parents, who provided their support for pursuing the research work.

REFERENCES

[1] Michael Miller, Cloud Computing, Web based Applications That Change the Way You Work and Collaborate Online, 1st printing in the United States of America, August 2008.

[2] Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, Cloud Computing: A Practical Approach, Mc Graw Hills, 2010.

[3] Judith Hurwitz, Robin Bloor, Marcia Kaufman, and Dr. Fern Halper, Cloud Computing For Dummies, Wiley Publishng Inc., 2010

[4] Ashraf B. El-Sisi, Medhat A. Tawfeek, Arabi E. keshk, Fawzy A. Torkey ."Intelligent Method for Cloud Task Scheduling Based on Particle Swarm Optimization Algorithm", International Arab Conference on Information Technology(ACIT 2014), pg-39-44, Dec 9-11, 2014.

[5] Kavita Bhatt and Dr. Mahesh Bundele, "Study and Impact of CloudSim on the run of PSO in Cloud Environment", International Journal of Innovations in Engineering and Technology(IJIET), Vol.2, Issue 4, august 2013.

[6] Suguna R, Divya Mohandass, Ranjani R,"A Noval Approach for Dynamic Cloud Partitioning and Load Balancing in Cloud Computing Environment", Journal of Theoretical and Applied Information Technology, Vol, 62, Issue 3, April 2014.

[7] S.N. Sivanandam, S.N. Deepa, Introduction to Genetic Algorithms, © Springer-Verlag Berlin Heidelberg 2008.

[8] R. Baskarane, T. Sendhil kumar, "Hybrid Optimization for Multiobjective Multicast Routing", International Journal of Research in Advent Technology, Vol.2, No.3, March 2014.