Indian Journal of Science and Technology, *Vol 7(5), 685–691, May 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Green Cloud: Heuristic based BFO Technique to Optimize Resource Allocation

Akshat Dhingra[1*] and Sanchita Paul[2]

[1]Department of Computer Science and Engineering, Birla Institute of Technology, Ranchi, India ; akshatdhingra@gmail.com

[2]Department of Computer Science and Engineering, Birla Institute of Technology, Ranchi, India ; sanchita07@gmail.com

Abstract

Cloud Computing is a relatively new technology and aims to offer "utility based IT services". Cloud Computing is now becoming increasingly popular because of the potential number of advantages that it aims to offer. However, with the growing popularity comes the increasing power consumption. Therefore, there is an utmost need to develop solutions that aim to save energy consumption without compromising much on the performance. Such solutions would also help reducing the costs thereby benefitting the cloud service providers. In this paper, an optimization technique called Bacterial Foraging has been used in order to continuously optimize the allocation of resources thereby improving the energy efficiency of the data centre. The results obtained after simulating a cloud computing environment and implementing the proposed algorithm make it clearly evident that cloud computing has great potential and offers significant performance gains as well as cost savings even under dynamic workload conditions.

Keywords: Allocation Optimization, Bacterial Foraging Optimization, Chemotaxis, Elimination, Energy Efficient Data Centres, Modified Best Fit Decreasing, Resource Power Consumption Minimization

1. Introduction

The designers have always primarily focused on improving the performance of computing systems and hence the performance has been steadily growing driven by more efficient system design and increasing density of the components described by Moore's law. Although the performance per watt ratio has been constantly rising, the total power draw by computing systems is hardly decreasing. On the contrary, it has been increasing every year that can be illustrated by the estimated average power usage across three classes of servers shown in the Table 1 in Watts/Unit.

Apart from the overwhelming operating costs due to high energy consumption, another rising concern is the environmental impact in terms of carbon dioxide (CO_2) emissions caused by this high energy consumption. The IT systems could be made friendlier to the environment either by making them more energy efficient or by finding a clean and a renewable source of energy to run these systems. The total amount of Greenhouse gas emissions caused by the IT industry that includes cell phones, computers and other IT equipment was 830 $MtCO_2$. This is 2% of the total emissions. Hence, the modern day computing not only requires being best in performance but also the most energy efficient. The IT industry is beginning to realise the fact that energy efficient systems would

Table 1. Power usage among three classes of servers

Class of Server	2000	2001	2002	2003	2004	2005	2006
Low-end	186	193	200	207	213	219	225
Mid-Range	424	457	491	524	574	625	675
High-end	5534	5832	6130	6428	6973	7651	8163

**Author for correspondence*

consequently cut down on the operating costs. We therefore make an effort to provide an energy efficient Cloud Computing environment on a software level making use of the virtualisation technology. Energy aware computing actually involves making the hardware as well as the software to work hand-in-hand in order to make the entire energy-efficient[1].

In this paper, an optimization technique called Bacterial Foraging has been used in order to continuously optimize the allocation of resources thereby improving the energy efficiency of the data centre. Section 1 briefly explains Cloud Computing, Section 2, tries to analyse the problem of modelling the power consumption and eventually provide a solution. Section 3 explains the proposed framework that is required in order to briefly understand the problem of making Cloud Computing energy efficient. The subsequent sections explain the simulation setup and analyse the results.

2. Modelling Power Consumption

IT equipment that consumes power consists of CMOS (Complex Metal Oxide Semiconductor) circuits. The power consumption by these circuits is divided into parts namely, Static power consumption and Dynamic power consumption. Static power consumption is the one that is done by the transistors and others components involved in a circuit and does not depend on the run time characteristics. Dynamic power consumption on the other hand varies particularly with some run time characteristics. In order to model this dynamic power consumption there has to be some method. The power consumption could either be measured by some instruments that are built-in or some software level tools. A detailed explanation has been given in a survey about the energy efficient cloud computing techniques by Dhingra et al.[2,3].

After a brief analysis of the power consumption, Fan et al.[4] found a noticeable relationship between them with the relationship being linear. This relationship can be expressed as:

$$P(u) = P_{idle} + (P_{busy} - P_{idle})^* u \quad (1)$$

Where P is the estimated power consumption; P_{busy} is the power consumed when the server is fully utilized; P_{idle} is the power consumed by the idle server; and u is the CPU utilization. The CPU utilisation changes depending on the workload.

In the recent times, there has been a growing increase in the usage of multi-core CPUs with high amount of memory. Hence, the memory begins to dominate the power consumption thereby making the relationship between the CPU utilisation and power consumption no longer linear. Hence it would be more accurate if a brief survey is done about the power consumption by the server of a particular configuration. This would make the eventual calculations rather more accurate.

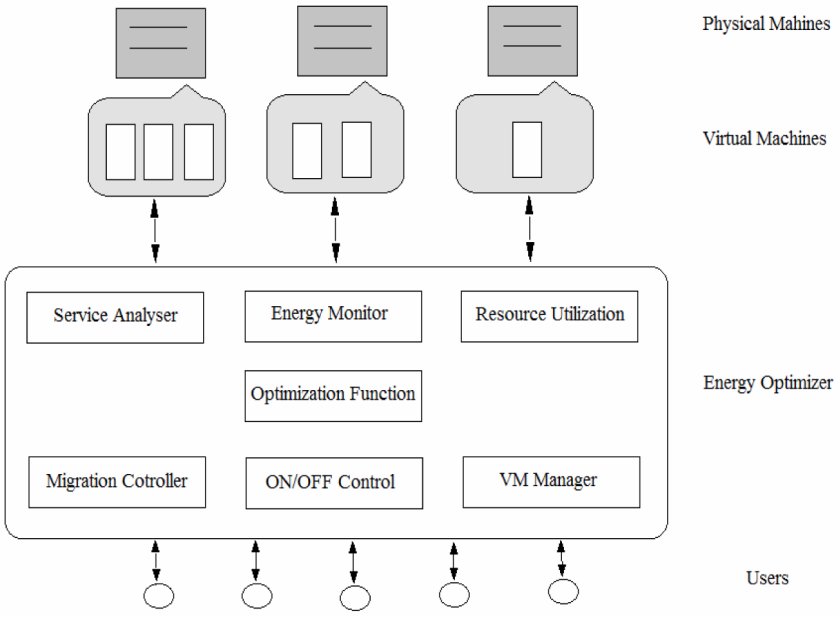

Figure 1. Framework for Energy Efficient Data Centre.

3. Framework

This paper aims to optimize the VM allocation in order to reduce the energy consumption. There are four major entities in the entire framework which are illustrated in the Figure 1.

The components in the energy optimization layer as shown in the below are:

- **Service Analyser:** Analyses the requirements of each submitted request and takes a decision whether to accept or reject the request based on the information gathered from the VM Manager, Energy Monitor and Resource utilization.
- **Energy Monitor:** Determines the energy consumed by each of the VMs.
- **Resource Utilization:** Determines the resources consumed by each of the Virtual Machines and hence has current information on the total resource consumption and availability.
- **Optimization Function:** This component tries to minimize the energy consumption without compromising much on the performance.
- **Migration Controller:** Performs the migration operation as and when commanded by the Optimization function and requires inputs from the VM manager.
- **ON/OFF Control:** Determines what VMs to turn ON or OFF as per the requirement
- **VM Manager:** Keeps a track of each VM including what physical machine it resides on and what are its resource requirements.

4. Optimizing Data Centre Resource Allocation using Heuristic based BFO

The problem of optimizing the VM allocation could be divided into two phases. In the first, VMs are placed on the hosts and in the second, the optimizations of the current allocations of the VMs take place. In the first step, we allocate the VMs to these physical hosts according to Modified Best Fit Decreasing (MBFD) Algorithm also taking into consideration the upper utilisation threshold of each host as proposed by Buyya et al.[5]. In this algorithm, the VMs are first sorted in a decreasing order of their CPU requirements and then each VM is allocated to a physical machine that provides least increase in the power consumption due to this allocation.

Now, in the second phase, resource allocation optimisation according to the current allocation takes place. This optimisation is done with the help of Bacterial Foraging Algorithm. In this, for the migrations of VMs from one host to another we make use of various heuristics. These heuristics used for performing optimum migrations are explained below.

- **Maximum Utilisation:** Perform the migration of that VM from the overloaded hosts that have the maximum CPU utilisation.
- **Minimum Utilisation:** Perform the migration of that VM from the overloaded hosts that have the minimum CPU utilisation.
- **Random Choice:** A VM to be migrated is selected on the basis of a uniformly distributed discrete random variable X, whose values index a set of VMs V_j allocated to a host j.

Bacterial Foraging Optimization Algorithm (BFO)[6] is belongs to the family of nature-inspired optimization algorithms. This optimization technique imitates the foraging behaviour of the E.Coli bacterium and involves four major steps. They are:

i. **Chemotaxis:** An E.Coli bacterium could either swim in a particular direction for some amount of time or it could tumble. It basically alternates between these two ways for its entire lifetime. Let us say, $\theta^i(j, k, l)$ represents i-th bacterium at j-th chemotactic, k-th reproductive and l-th elimination-dispersal step. C(i) is the step isze taken during the tumble (run length unit) mode. Then mathematically, the movement of the bacterium may be represented by

$$\theta^i(j+1,k,l)=\theta^i(j,k,l)+C(i)\Delta(i)/\sqrt{(\Delta^T(i)\Delta(i))} \quad (2)$$

In the above equation Δ indicates a vector in the random direction whose elements lie in [-1,1].

ii. **Swarming:** When the cells are moving along a positive nutrient gradient, they begin to release a chemical called aspartate that attracts the other bacteria and hence form patterns of swarms. This cell-to-cell signalling is represented by the following function

$$J_{cc}(\theta, P(j,k,l)) = \Sigma_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l)) = \Sigma_{i=1}^{S} [-d_{attractant} \exp (-w_{attractant} \Sigma_{m=1}^{p} (\theta_m - \theta_m^i)^2)] + \Sigma_{i=1}^{S} [-h_{repellant} \exp (-w_{repellant} \Sigma_{m=1}^{p} (\theta_m - \theta_m^i)^2)] \quad (3)$$

Here Jcc(θ, P(j,k,l)) is the objective function value to be added to the actual objective function (to be minimized) to present a time varying objective function, S is the total number of bacteria, p is the number of variables to be optimized. $d_{attractant}$, $w_{attractant}$, $h_{repellant}$, $w_{repellant}$ are different coefficients that should be chosen properly.

iii. **Reproduction:** The healthy bacterium survive and split into two and the least healthy ones eventually die.

iv. **Elimination and Dispersal:** Due to gradual or sudden changes in the local environment certain bacteria might get killed. Hence, certain bacteria get killed and the ones that survive are dispersed to a new location.

The resource allocation optimization that takes place periodically every fixed interval of time happens in the following manner:

- All the servers that have CPU utilization beyond the threshold are determined.
- All set of possible migrations (solutions) are determined.
- These solutions are determined for fitness first and then evaluated in terms of the overall power consumption.
- The solution that consumes the minimum amount of energy is selected and the others are rejected.

The pseudo-code for the optimization algorithm which is the second step that optimizes the current allocation is explained below.

4.1 Resource Allocation Optimization using BFO

Input: Number of VMs and physical machines, minimum and maximum resource requirements of each VM.

Output: Best Solution.

1. Population =: All feasible solutions;
2. For (l = 0 to N_{ed}) //Elimination-Dispersal loop.
   For (k = 0 to N_{re}) //Reproduction loop
   For (j = 0 to N_c) // Chemotaxis loop
   ChemotaxisandSwim(Population,N_s); //Searches for the best solution.
   For (Each Cell in Population)
   If (PowerConsumed(Cell) <=P owerConsumed ($Cell_{best}$))
   $Cell_{Best}$: =Cell; //Solution that consumes // minimum amount of power.
   End
   End
   End
3. SortByHealth(Population);//Sorting done in increasing order of power //consumption
4. Selected=: SelectedByHealth(Population); //Selection of the best solution.
5. Population=: Selected;
   End
   End

Chemotaxis and Swim (Population)

Begin
For (Each cell in Population)
{

1. Determine the current resource usage and the power consumed by the cell; PowerConsumed (Cell);
2. Determine the VM to be migrated from the overloaded hosts according to a heuristic (Minimum Utilisation, Maximum Utilisation, Random Choice);
3. Migrate the VM to the host that offers the least increase in the power consumption after VM allocation.
4. Determine the power consumed by the host if the migration is done (NewPower);
5. If (NewPower< PowerConsumed(Cell))

Make necessary modifications in the cell and store;
}
End

Here, the parameters N_c, N_{re}, and N_{ed} represent the number of Chemotaxis, Reproduction and Elimination steps respectively. The values of N_c, N_{re}, N_{ed} depends on the level of optimization that is desired and may vary in different scenarios. Hence, it would be interesting to analyse the result with different values of N_c. A 'Cell' represents each possible solution and '$Cell_{Best}$' is the best solution after each chemotaxis step. 'PowerConsumed' is a function that determines the power consumed by each Cell. 'N_s' represents the total number of swim steps.

The flowchart for resource allocation optimization using bacterial foraging is shown in Figure 2.

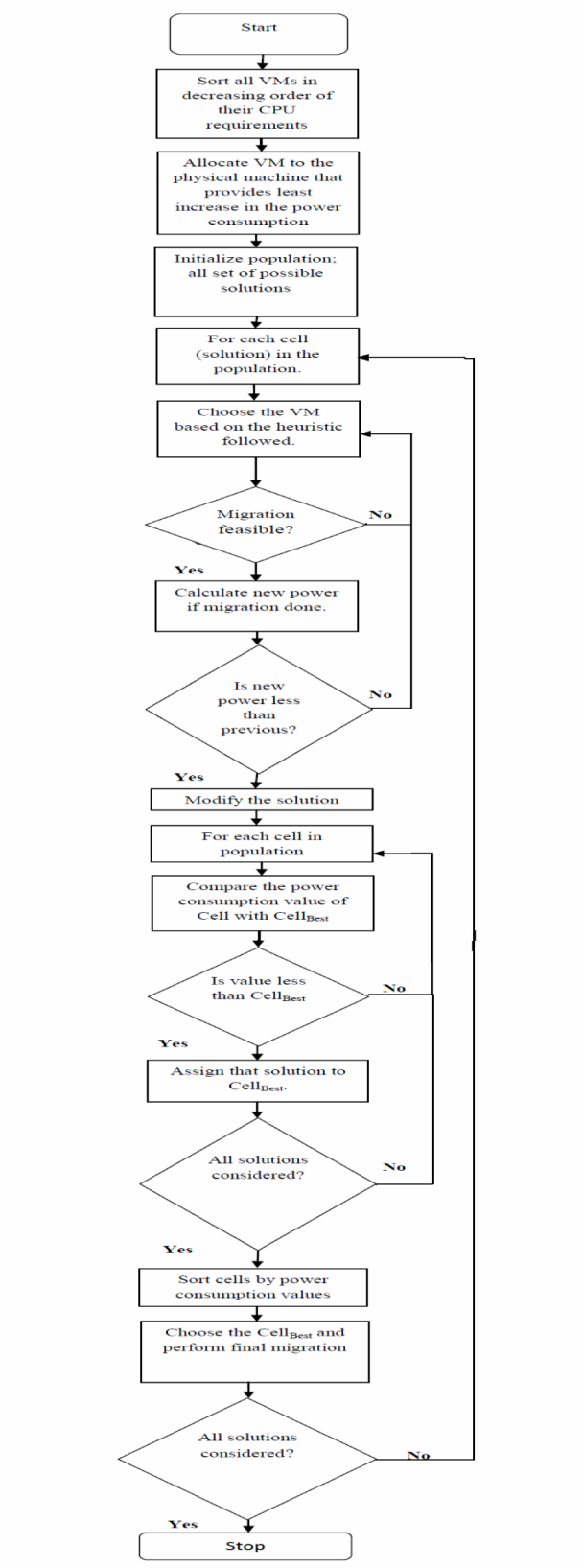

Figure 2. Flowchart showing the Allocation Optimization Process.

5. Implementation and Result Analysis

After proposing the optimisation algorithm, it becomes necessary to test the performance of the same. We hence chose to evaluate the proposed technique by implanting it in a simulated cloud computing environment. Therefore CloudSim toolkit was used for implementation and evaluation. 50 servers and 50 VMs were simulated whose characteristics are described in the Tables 2, 3 and 4 [5,7].

Each VM gets the workload which is random in nature and varies with a uniformly distributed random variable. This is done because it is difficult to simulate the workload of an arbitrary application.

For the sake comparison, the results were compared with a Non-Power-Aware Policy that applies no optimisations and the CPU always consumes the maximum amount of power. The results obtained after a series of experiments are shown in Figures 3 and 4.

From Figures 3 and 4, we observe a strong relationship between the utilisation threshold and the energy consumption. There is a decrease in the energy consumption with increase in the utilisation threshold. But there is also a trade-off with the SLA violations as they increase with increase in the upper utilisation threshold. The results have been summarised in the table below and for the purpose of comparison, the results obtained after implementing Non Power Aware (NPA) and Dynamic Voltage Frequency Scaling (DVFS) policies in a simulated cloud computing environment have also been included.

From Table 5 results it becomes apparent that by optimizing the allocation of VMs according to the current CPU utilisation, the energy consumption could be significantly reduced only at the cost of some SLA violation. This therefore makes the proposed techniques suitable in scenarios where energy savings and hence profit maximisation is a greater concern and SLA violations are acceptable to some extent.

Table 2. Details about the Simulated Servers

Server	Processor	Cores	MIPS	RAM	Hard Disk
HP ProLiant G4	Intel Xeon 3040	2	1860	4	1 TB
HP PorLiant G5	Intel Xeon 3075	2	2660	4	1 TB

*MIPS: Million Instructions per Second

Table 3. Power Consumption by the Simulated Servers at different levels of CPU utilisation in Watts[7]

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

Table 4. VM Characteristics

Type	MIPS	RAM	Hard Disk
1	2500	870 MB	2.5 GB
2	2000	1740 MB	2.5 GB
3	1000	1740 MB	2.5 GB
4	500	613 MB	2.5 GB

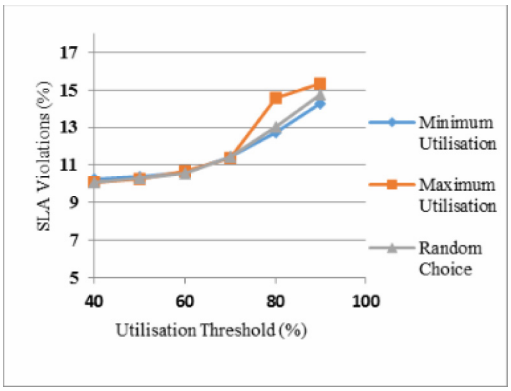

Figure 3. Power Consumption under various heuristics.

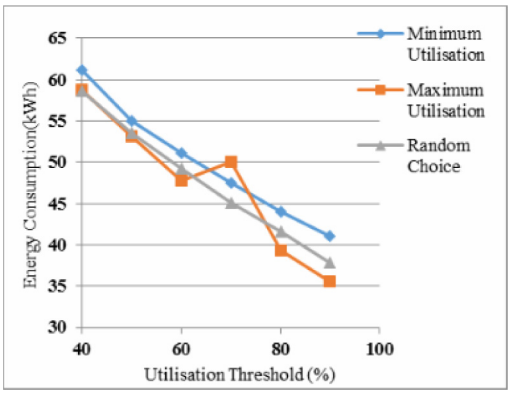

Figure 4. SLA violations under various heuristics.

6. Limitations and Future Directions

A limitation in the proposed solution is that the requirements should be known in advance for initial allocation to take place. This information may not be known in certain scenarios. Also, the cost of migration has not been considered as it is assumed that all the machines are located in the same location. This however may not always be the case. The results are obtained with the help of simulation and it would be interesting to analyse the results if the algorithm is implemented in a real scenario. As a part of the future work we would also suggest exploring some hybrid optimization techniques that harvest the benefits of Bacterial Foraging with Genetic Algorithms or Greedy Knapsack combined with better set of heuristics.

Table 5. Simulation Results

Policy	Energy(kWh)	VM Migration	Average SLA %
NPA	150.68	0	0
DVFS	52.98	0	0
Minimum Utilisation 40-90%	50.02	5923	11.61
Maximum Utilisation 40-90%	46.2	4956	12.04
Random Choice 40-90%	47.6	5045	11.7

7. References

1. Minas L, Ellison B. Energy efficiency for information technology: how to reduce power consumption in servers and data centers. Intel Press; 2009.
2. Dhingra A, Paul S. Green cloud: smart resource allocation and optimization using simulated annealing technique. Indian Journal of Computer Science and Engineering. 2014 Apr–May; 5(2):40–48.
3. Dhingra A, Paul S. A survey of energy efficient data centers in a cloud computing environment. International Journal of Advance Research in Computer and Communication Engineering. 2(10); 2013 Oct; 4033–40.

4. Fan X, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA); 2007; ACM New York, NY, USA. p. 13–23.
5. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency Comput Pract Ex. 2012; 24(13):1397–420.
6. Das S, Biswas A, Dasgupta S, Abraham A. Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. Foundations of Computational Intelligence. 3. Studies in Computational Intelligence. 2009; 203:23–55.
7. Available from http://www.specpowerbenchmark.org