Improve Energy Consumption Model for Cloud Data Centers using PSO Algorithm

C. Thirumalai Selvan[1]
[1]Assistant Professor
Department of CSE K.S.R. College of Engineering,Tiruchengode,India

M. Santhoshkumar[2], R M. Santhoshkumar[3], S. Sasidharan[4],
[2,3,4]Final Year, Department of CSE,
K.S.R. College of Engineering, Tiruchengode, India

***Abstract*—The growth of cloud computing has resulted in uneconomic energy consumption, which has negatively impacted the environment through the generation of carbon emissions. This project proposes a distributed Locust-inspired scheduling algorithm to reduce cloud computing consumed energy (LACE). It schedules and optimizes the allocation of virtual machines (VMs) based on behavior derived from locusts. LACE distributes scheduling among servers; each server is responsible for allocating and migrating its VMs. Hence, the scheduling load is distributed between servers rather than being centralized in one component. LACE was thoroughly evaluated by comparing it with long-standing VM scheduling algorithms: dynamic voltage–frequency scaling (DVFS),energy-aware scheduling using the workload-aware consolidation technique, and the static threshold with minimum utilization policy. In addition, this project proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) and Platform as services clouds (PaaS). This project presents an algorithm based on the Superior Element Multitude Optimization (SEMO), which aims to minimize the overall workflow execution cost while meeting deadline constraints. The main scope of the project is used to analyze best available resource in the cloud environment depends upon the total execution time and total execution cost which is compare between one process to another process. If the provider satisfies the time least time, then the process becomes to termination.**

Keywords—Cloud Computing, Energy Efficiency, Resource Management, Virtualization

I. INTRODUCTION

Cloud computing is internet-based computing in which large groups of remote servers are networked to allow sharing of data-processing tasks, centralized data storage, and online access to computer services or resources. Clouds can be classified as public, private or hybrid. Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications.

The main enabling technology for cloud computing is virtualization. Virtualization software allows a physical computing device to be electronically separated into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks. Cloud computing adopts concepts from Service oriented Architecture (SOA) that can help the user break these problems into services that can be integrated to provide a solution. Cloud computing provides all of its resources as services, and makes use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way.

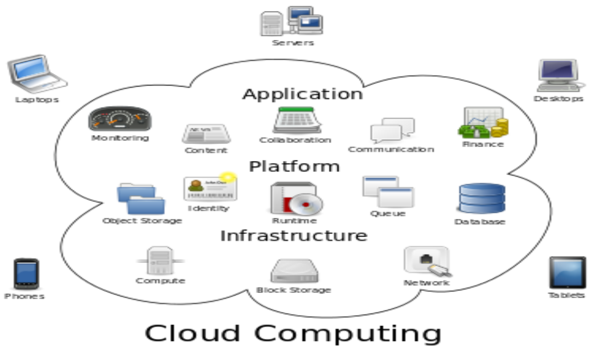


Fig (1.1) CLOUD COMPUTING ARCHITECTURE

Cloud computing is a kind of grid computing; it has evolved by addressing the QOS (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional parallel computing techniques.

A. *Characteristics*

Cloud computing exhibits the following key characteristics

- Agility
- Application programming interface
- Cost reductions
- Device and location independence
- Maintenance
- Multi tendency
- Performance
- Productivity
- Security

The main objectives of this thesis are (i) To reduce energy consumption in Cloud Datacenters ii) To make the system adaptable in situations where multiple initial set of resource availability iii) To make the environment suitable for multiple cloud service providers iv) To reduce data transfer cost between different cloud service providers v) To exhibit significant levels of fault tolerance under heavy workloads.

II.RELATED WORKS

In the paper [1] 'The NIST Definition of Cloud Computing' the authors described Cloud computing as an evolving paradigm. The NIST definition [1] characterizes important aspects of cloud computing and is intended to serve as a means for broad comparisons of cloud services and deployment strategies, and to provide a baseline for discussion from what is cloud computing to how to best use cloud computing. The service and deployment models defined form a simple taxonomy that is not intended to prescribe or constrain any particular method of deployment, service delivery, or business operation.

The intended audience of this document is system planners, program managers, technologists, and others adopting cloud computing as consumers or providers of cloud services. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

A. Essential Characteristics:

a. On-demand self-service:

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

b. Broad network access:

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

c. Resource pooling:

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

In the paper [2] 'A particle swarm optimization for reactive power and voltage control considering voltage stability' the authors described about particle swarm optimization for reactive power and voltage controlconsidering voltage stability. The proposed method determines a control strategy with continuous and discrete control variables such as AVR operating values, OLTC tap positions, and the amount of reactive power compensation equipment. The method also considers voltage stability using a continuation power flow technique. The feasibility of the proposed method is demonstrated on model power systems with promising results [2].

Reactive power and voltage Control (Volt/Var Control: VVC) determines an on-line control strategy for keeping voltages of target power systems considering varying loads in each load point and reactive power balance in target power systems. Conventionally, VVC is usually realized based on power flow sensitivity analysis of the operation point considering execution time and available data from the actual target power system.

Recently, voltage stability problem has been dominating and the consideration of the stability has been required in VVC problem. Since fast computation of voltage stability is required for VVC, continuation power flow (CPFLOW)[3] is suitable for the calculation. The authors has been developed a practical CPFLOW and verified it with an actual power system [4].

VVC can be formulated as a mixed-integer nonlinear optimization problem with continuous state variables such as AVR operating values and discrete state variables such as OLTC tap positions and the amount of reactive power compensation equipment. The objective function can be varied according to the power system condition. For example, the function can be loss minimization of the target power system for the normal operating condition. Conventionally, the methods for VVC problem have been developed using various methods such as fuzzy, expert system, mathematical programming, and sensitivity analysis.

However, a practical method for a VVC problem formulated as a mixed-integer nonlinear optimization problem has been eagerly awaited. Particle swarm optimization (PSO) is one of the evolutionary computation (EC) techniques [5]. The original method is able to handle continuous state variables easily and search a solution in a solution space efficiently.

However, the method can be expanded to treat both continuous and discrete variables. Therefore, the method can be applicable to a VVC problem. This paper presents a PSO for a VVC problem formulated as a mixed integer nonlinear optimization

In the paper [6] 'a dynamic critical path algorithm for scheduling scientific workflow applications on global grids' the authors stated that effective scheduling is a key concern for the execution of performance driven Grid applications. In this paper, to propose a Dynamic Critical Path (DCP) based workflow scheduling algorithm that determines efficient mapping of tasks by calculating the critical path in the workflow task graph at every step. It assigns priority to a task in the critical path which is estimated to complete earlier. Using simulation, It have compared the performance of the proposed approach with other existing heuristic and meta-heuristic based scheduling strategies for different type and size of workflows[6]. The results demonstrate that DCP based approach can generate better schedule for most of the type of workflows irrespective of their size particularly when resource availability changes frequently.

Many of the large-scale scientific applications executed on present-day Grids are expressed as complex e-Science workflows. A workflow is a set of ordered tasks that are linked by data dependencies.

A Workflow Management System (WMS) [7] is generally employed to define, manage and execute these workflow applications on Grid resources. A WMS may use a specific scheduling strategy for mapping the tasks in a workflow to suitable Grid resources in order to satisfy user requirements. Numerous workflow scheduling strategies have been proposed in literature for different objective functions [8].

In the paper [10] ‘A budget constrained scheduling of workflow applications on utility grids using genetic algorithms’ the authors stated that over the last few years, Grid technologies have progressed towards a service-oriented paradigm that enables a new way of service provisioning based on utility computing models. Users consume these services based on their QOS (Quality of Service) requirements.

In such “pay-per-use” Grids, workflow execution cost must be considered during scheduling based on users’ QoS constraints. In this paper, they propose a budget constraint based scheduling[10], which minimizes execution time while meeting a specified budget for delivering results.

A new type of genetic algorithm is developed to solve the scheduling optimization problem and test the scheduling algorithm in a simulated Grid tested.

Utility computing [11] has emerged as a new service provisioning model [12] and is capable of supporting diverse computing services such as servers, storage, network and applications for e-Business and e-Science over a global network. For utility computing based services, users consume the services when they need to, and pay only for what they use.

With economy incentive, utility computing encourages organizations to offer their specialized applications and other computing utilities as services so that other individuals/organizations can access these resources remotely.

III. METHODOLOGY

A. *Problem Definition*

The presented process of an algorithm should estimates the optimal number of resources that need to be leased so that the execution cost of a workflow is minimized. Their algorithm should generate a task to resource mapping and is designed to run online. The schedule and resources are updated every charge time interval (i.e., every hour) based on the current status of the running virtual machine (VMs) and tasks.

The approach should take advantage of the elasticity of the cloud resources but fails to consider the heterogeneous nature of the computing resources by assuming there is only one type of VM available.

B. *Overview of thesis*

The cloud computing model leverages a vast number of virtualized computing resources to provide them as utilities to customers on a pay-as-you-go basis. The growth of cloud computing has resulted in uneconomic energy consumption, which is harmful to the environment due to the huge carbon footprints of cloud datacenters. A typical datacenter consumes as much energy as 25,000 households. An average datacenter can also produce more than 150 million metric tons of carbon annually. In 2013, US datacenters used an estimated 91 billion kilowatt-hours of electricity. Therefore, green cloud computing must be used to protect the environment. Green cloud computing aims to save the environment from datacenters’ carbon emissions by reducing energy consumption levels.

One of the most important issues for green cloud environments concerns where to place new virtual machine (VM) requests across physical servers in a way that ensures reduced energy consumption. However, the main drawbacks of met heuristics concern their steep exponential running periods and high levels of power consumption, rendering them advisable to use only when no problem-dependent heuristic is available.

So this thesis considers a decentralized software optimization approach to ensuring robustness, scalability, and cost effectiveness when designing the Locust-inspired scheduling Algorithm to reduce energy consumed in Cloud datacenters (LACE).

Locusts transition between two phases: solitary and gregarious. During the gregarious phase, locusts display gluttonous behavior; each insect can eat ten times its normal food intake.

Sometimes, locusts even cannibalize weaker locusts. In contrast, in the solitary phase, locusts behave in the opposite manner: they eat only grass and only when they are hungry [7].

LACE imitates locust behavior under these two phases to greedily consolidate VMs into fewer servers and it then switches off idle servers. In addition, dead line resource provisioning is also carried out for better cloud usage.

- The project considers fundamental features of IaaS providers such as the dynamic provisioning and heterogeneity of unlimited computing resources.
- To achieve this, both resource provisioning and scheduling are merged and modeled as an optimization problem.
- PSO is then used to solve such problem and produce a schedule defining not only the task to resource mapping, but also the number of nodes to be assigned.
- The process referred in the single cloud provider is used to compute the consumption time and execution cost for running the process in the environment.
- The scheduling process is done in the basis of set of resources, number of task which are defined to that resource in the environment.
- To compute the result of total consumption cost and total execution time PSO logic is used.

C. *SYSTEM MODEL*

This paper focuses on the problem of scheduling VMs on a set of physical servers to minimize energy consumed in a datacenter by reducing the number of running servers used. It proposed a distributed Locust-inspired scheduling algorithm to reduce cloud computing consumed energy (LACE). LACE schedules and optimizes the allocation of virtual machines (VMs) based on behavior derived from locusts. LACE distributes scheduling among servers; each server is responsible for allocating and migrating its VMs.

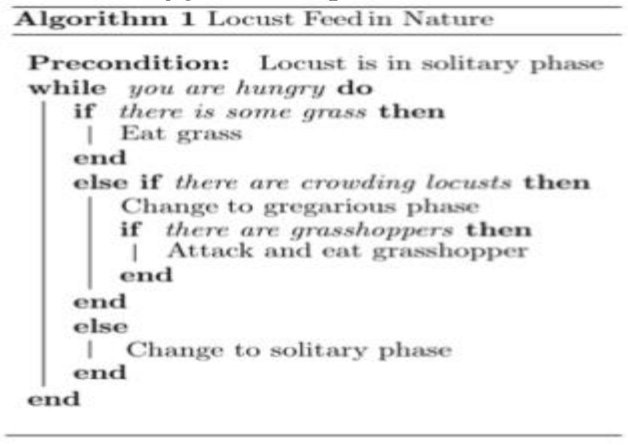
The study and its practical application are split into following area.

- Locust inspired scheduling algorithm
- Cloud providers addition
- Resources addition
- Process addition
- Assign process/resource
- Tasks addition
- PSO-(ESO)

a. *Locust Inspired Scheduling Algorithm*

Locusts exhibit interestingly Flexible behavior in that they can transition between two opposing phases (a solitary phase and a gregarious phase). Locusts typically live in the solitary phase, eating grass when hungry until the population grows and each locust feels a crowd of locusts around it. At this point, the insects begin to enter the gregarious phase. In this phase, each locust becomes gluttonous; it feeds excessively not only on grass but also on weaker locusts. Then, as the population density falls, each locust returns to its solitary phase.

In this module, the algorithm is executed out as follows. The precondition is “locust is in solitary phase”. Then it executes with a loop with the condition as “when the locust is in hungry state”, if there is some grass then eat the grass, else if there are crowded locusts then change to gregarious phase and if there are any grasshoppers, attack and eat grass hoppers. If grass hoppers quantity reduced then switch to solitary phase. The algorithm is shown below.



```
Algorithm 1 Locust Feed in Nature

Precondition:  Locust is in solitary phase
while  you are hungry do
    if  there is some grass then
        Eat grass
    end
    else if there are crowding locusts then
        Change to gregarious phase
        if  there are grasshoppers then
            Attack and eat grasshopper
        end
    end
    else
        Change to solitary phase
    end
end
```

Fig (3.1) LACE BEHAVIOR

b. *CLOUD PROVIDERS*

This module is used to add the cloud provider details to the database table. The cloud provider id and the cloud provider name are added to the table. All the record details can be viewed using the Grid View control in a form. The details are store to ‘Cloud Providers’. The resource details must include which cloud provider id it belongs to.

c. *RESOURCES*

This module is used to add the resource details to the database table. The resource id and the resource name and cloud provider id are added to the table. All the record details can be viewed using the Grid View control in a form. The details are store ‘Resources’ table. The cloud provider ids are fetched from the ‘Cloud Providers’ table and any one id is selected as resource type for this record.

d. *PROCESS*

This module is used to add the process details to the database table. The process id and the process name are added to the table. All the record details can be viewed using the Grid View control in a form. The details are store the ‘Processes’ table. The details regarding which process use which resource is added later. The task details contain which process it belongs to.

e. *ASSIGN PROCESS/RESOURCE*

This module is used to add the process/resource details to the database table. The process resource id (used as primary key), process id and the resource id are added to the table. All the record details can be viewed using the Grid View control in a form. The ‘Process Resource’ table is used to store the records. The details regarding which process use which resource is assigned here.

f. *TASKS*

This module is used to add the task details to the database table. The task id and the task name and process id, resource id and time taken in that resource are added to the table. All the record details can be viewed using the Grid View control in a form. The ‘Task’ table is used to store the records. The process ids are fetched from the ‘Processes’ table and any one id is selected as process id for this record. The relationship between process id and task id is one-to-many relationship.

g. *PSO(ESO)*

The scheduling process is done in the basics of the set of resources, number of task which are defined to that resource in the environment .To compute the result of total consumption cost and total execution time PSO logic is used

1. *EXECUTION TIME MATRIX GENERATION*

This module generates the execution time matrix in which number of resources is taken as columns and tasks are taken as rows and the time the tasks taken to complete in those resources are stored as values.

2. *TRANSFER TIME MATRIX GENERATION*

This module generates the transfer time matrix in which number of taken are taken as columns and rows (square matrix is prepared) and the time a task transfers the data to other task is stored as values. So the diagonal elements are always zero since same task has no data transfer operation.

3. *SCHEDULE GENERATION*

Initially, the set of resources to lease R and the set of task to resource mappings M are empty and the total execution cost TEC and time TET are set to zero. After this, the algorithm estimates the execution time of each workflow task on every resource r_i $R_{initial}$. This is expressed as a matrix in which the rows represent the tasks, the columns represent the resources and the entry Exe $Time_{i, j}$

represent the time it takes to run task ti on resource r_j. This time is calculated using Fig a.

The next step is the calculation of the data transfer time matrix. Such matrix is represented as a weighted adjacency matrix of the workflow DAG (Directed acyclic graph) where the entry $TransferTime_{i,j}$ contains the time it takes to transfer the output data of task t_i to task t_j. This value is taken from database and is zero whenever ij or there is no directed edge connecting t_i and t_j. An example of these matrices is shown in Figure a) and b) below.

$$exeTime = \begin{array}{c|ccc} & r_1 & r_2 & r_3 \\ \hline t_1 & 2 & 1 & 4 \\ t_2 & 4 & 3 & 6 \\ t_3 & 10 & 6 & 15 \\ t_4 & 7 & 4 & 12 \\ t_5 & 8 & 4 & 10 \\ t_6 & 3 & 2 & 7 \\ t_7 & 12 & 7 & 18 \\ t_8 & 9 & 5 & 20 \\ t_9 & 13 & 8 & 19 \end{array}$$

(a)

Fig (3.2) Matrix representation of execution time

$$transferTime = \begin{array}{c|ccccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \\ \hline t_1 & 0 & 9 & 9 & 9 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ t_3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ t_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ t_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ t_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ t_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ t_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

(b)

Fig (3.3) Matrix representation of transfer time

IV CONCLUSION

The study shows that LACE improved resource utilization levels to a much greater degree than two other benchmarks by reducing the number of active servers involved. In addition, it extends the resource model to consider the data transfer cost between data centers so that nodes can be deployed on different regions. Extending the algorithm to include heuristics that ensure a task is assigned to a node with sufficient memory to execute it will be included in the algorithm.

Also, it assigns different options for the selection of the initial resource pool. For example, for the given task, the different set of initial resource requirements is assigned. In addition, data transfer cost between data centers are also calculated so as to minimize the cost of execution in multi-cloud service provider environment.

The main contribution of this study is

- Adaptable in situations where multiple initial set of resource availability.
- Suitable for multiple cloud service provider environments.
- Data transfer cost is reduced between different cloud data centers.

The system is very flexible and user-friendly, so the maintenance based on the changing environment and requirements can be incorporated easily. Any changes that are likely to cause failures are prevented with security and preventive measures could be taken.

The coding is done in understandable and flexible method program which helps easy changing. Since MS-SQL Server and.NET are very flexible tools, user can easily incorporate any modular program in the application.

REFERENCES

[1] P. Mell, T. Grance, “The NIST definition of cloud computing-recommendations of the National Institute of Standards and Technology” Special Publication 800-145, NIST, Gaithersburg, 2011.

[2] J. Yu and R. Buyya, “Workflow Scheduling Algorithms for Grid Computing”, Tech. Rep., GRIDS-TR-2007-10, University of Melbourne, Australia.

[3] M. Rahman, S. Venugopal, and R. Buyya, “A dynamic critical path algorithm for scheduling scientific workflow applications on global grids,” in Proc. 3rd IEEE Int. Conf. e-Sci. Grid Comput., 2007, pp. 35–42

[4] J. Yu and R Buyya, “A budget constrained scheduling of workflow applications on utility grids using genetic algorithms,” in Proc. 1st Workshop Workflows Support Large-Scale Sci., 2006, pp. 1–10.

[5] J. Yu and R. Buyya, “Taxonomy of Workflow Management Systems for Grid Computing”, Journal of Grid Computing, 3(3-4): 171-200, Springer, New York, USA, Sept. 2005.

[6] T. Eilam et al., “A utility computing framework to develop utility systems”,IBM System Journal, 43(1):97-120, 2004.

[7] G. Thickins, “Utility Computing: The Next New IT Model”, Darwin Magazine, April 2003.

[8] Y. Fukuyama and Y. Nakanishi, “A particle swarm optimization for reactive power and voltage control considering voltage stability,” in Proc. 11th IEEE Int. Conf. Intell. Syst. Appl. Power Syst., 1999, pp. 117–121.

[9] H. Yoshida, Y. Fukuyama, et al., "Practical Continuation Power Flow for Large-Scale Power System Analysis", Proc. of IEE of Japan Annual Convention Record, No. 1313, 1998 (in Japanese).

[10] H. D. Chiang, et al., "CPFLOW: A Practical Tool for Tracing Power System Steady-State Stationary Behavior Due to Load and Generation Variations", IEEE Trans. on Power Systems, Vol. 10, No. 2, May 1995.

[11] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proc. of IEEE International Conference on Neural Networks, Vol. IV, pp.1942-1948, Perth, Australia, 1995

[12] S. Kim, and J. Browne, “A General Approach to Mapping of Parallel Computation upon Multiprocessor Architectures”, Proceedings of IEEE International Conference on Parallel Processing, 1988, IEEE press.