Journal of Computer Science

Original Research Paper

Location-Aware Energy-Efficient Workload Allocation in Geo Distributed Cloud Environment

[1]Soha Rawas and [1,2]Ahmed Zekri

[1]Department of Mathematics and Computer Science, Faculty of Science, Beirut Arab University, Lebanon
[2]Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Egypt

Article history
Received: 16-11-2017
Revised: 02-02-2018
Accepted: 13-03-2018

Corresponding Author:
Soha Rawas
Department of Mathematics and Computer Science, Faculty of Science, Beirut Arab University, Lebanon
Email: rawassoha@gmail.com

Abstract: The proliferation of cloud computing relied on the virtualization of the compute and storage resources and provisioning them dynamically according to users' needs on a pay-per-use model. Massive cloud providers have geo-distributed cloud data centers to ensure service reliability, availability and satisfy user's need. Therefore, cloud management systems are necessary to increase the profit of cloud providers and to improve the quality-of service demanded by users. This paper focuses on an energy-efficient method to solve the problem of allocating data-intensive workloads in geographically distributed data centers. The workload's tasks are characterized by large data transfer times than their execution times. The problem formulated as a nonlinear programming optimization problem. Then, to find an optimal solution to the problem, meta-heuristic genetic algorithm is proposed. The designed heuristic takes into account the cost of the data transfer time from the storage location to the compute servers as well as the workload makespan on the available hosts. Extensive simulations using the CloudSim simulator are conducted to evaluate the efficacy of the proposed allocation method and how it performs with respect to other methods in the literature. Our results show significant enhancements in energy consumption while respecting the user's QoS.

Keywords: Green Computing, Energy Efficiency, Geo-Distributed Data Centers, Genetic Algorithm

Introduction

Throughout its history, computer systems have evolved in a spiral way of integration and distribution. They experienced a transition from centralized, massive, shared mainframes in the 1970s to decentralized, handy, personal PCs in the 1990s. In 2010, nevertheless, they moved into consolidated and shared virtualized computing, networking and storage resources invisible to the users, or the so-called cloud computing systems (Fox *et al*., 2009).

Cloud computing has developed into a widespread computing paradigm which provides systems with cheaper and accessible resources. It also provides researchers with a new method to set up their required computing systems and carry out their scientific as well as business applications without purchasing infrastructure or software at all (Fox *et al*., 2009). However, using cloud services allow for a more flexible and pay-as-yougo pattern. Moreover, it becomes like conventional utilities in everyday life (e.g., water, electricity, gas and telephony) (Buyya *et al*., 2009).

The extensive growth of cloud systems has led to the construction of geo-distributed Data Centers (DCs) worldwide with thousands of computing, networking and storage nodes. Consequently, this led to a drastic increase in the DCs energy consumption, that directly affect cloud providers profit and leads to serious environmental issues (high carbon emission) that affect cloud computing sustainability.

According to a study by the Uptime Institute and McKinsey (2008), server clusters in DCs contribute to 30% of the world's CO2 emissions and will surpass those of the airline industry by 2020. Other recent studies by Greenpeace indicate that IT carbon footprints occupy 2% of the global greenhouse gas emissions (Sverdlik, 2011; Cook, 2012), which equates the CO2 emissions by the aviation industry (Gartner, 2007), leading to the drastic greenhouse effect. Koomey expected in (2007) that energy consumptions in DCs would continue to increase rapidly unless advanced energy efficient mechanisms are established and applied. The most recent studies by Greenpeace estimate the

number of currently online people to be around 2.5 billion. They believe that this number is expected to increase by 60% in the next 5 years. That means about 80% of the planet adult population will be connected (Cook, 2014; Cook and Pomerantz, 2015). Thus, achieving power-efficiency in today's Internet and cloud servers is a fundamental concern.

To tackle the issue of high-energy usage, eliminating electricity ineffectiveness and waste in the way it is carried to computing resources is a must. It is also important to consider how these resources are used to serve the user applications workloads. Energy-aware software management techniques have the potential to reduce energy consumption in cloud DCs between 30 and 90% (CGS, 2015). These techniques cover resource virtualization, process and data migrations, resource consolidations and user applications scheduling techniques.

Cloud service providers distribute their DCs in different geographical regions to ensure availability and disaster recovery. Managing data movement between the computing and storage nodes is crucial for delivering good quality of service. Therefore, selecting proper DCs to store applications data and keeping it near to the processing nodes recently attracted researchers' attention.

The allocation of tasks (called cloudlets (A Cloudlet is defined as an application task running on a cloud environment. It is characterized by a set of attributes such as: length, type, storage, memory requirements, among others) mapping in cloudSim terminology) within a batch of data-intensive workload to Virtual Machines (VMs) in a distributed cloud environment, which respond to thousands of user's requests during a short period of time, is a challenging problem. Therefore, a proper allocation method allocates the workload tasks to geographically distributed cloud DCs taking into account the cost of the network delay can substantially reduce the overall execution time of the workload' last (or the makespan) and optimize the utilization of the VMs compute resources. Consequently, an improvement of the overall system performance, as well as the user's QoS, is expected.

This paper studies the problem of how to consider the delay factor when designing an energy efficient task allocation algorithm in a geo-distributed cloud environment. The goal of the proposed method is to minimize the total execution cost of data-intensive workloads in geographically distributed compute resources owned by one cloud service provider such as Google and Amazon. It aims to increase the QoS and hence customer satisfaction, as well as to reduce the data centers' energy consumption using the Dynamic Voltage and Frequency Scaling (DVFS) technique. The proposed allocation method evaluated using a multicloud framework developed on the CloudSim (Calheiros *et al*., 2011) simulator. The reason for evaluating the proposed method on a simulator rather than real cloud is that our method requires different scenarios with various infrastructure properties to evaluate its efficacy in terms of QoS and energy efficiency. Our contribution in this study can be summarized as follows:

- The problem of allocating data-intensive bagof-tasks BoT workloads is formulated as a nonlinear optimization problem taking into consideration the network's delay times to minimize the makespan of the workload, i.e., enhance the users QoS
- A meta-heuristic genetic algorithm is designed to solve the proposed optimization problem on geographically distributed data centers
- Extensive simulations using CloudSim tool are conducted to validate the proposed algorithm using both real and synthetic workload traces. The simulation results showed the superiority of the LAEE algorithm in enhancing the user's QoS as well as reducing the energy consumption of the data centers compared to other competing algorithms

The organization of this paper is as follows. Section 2 gives some of the related works. Section 3 presents the assumptions of the cloud system model. Section 4 presents our mathematical formulation. Section 5 presents the proposed genetic algorithm to find a nearoptimal solution to the formulated optimization problem. Section 6 details the results obtained using a simulation toolkit and conclude the paper in Section 7.

Related Works

The proliferation of cloud computing has led to the establishment of distributed systems to ensure availability and improve the processing of user requests. Therefore, the mapping process of cloud user's workloads to resources of globally distributed DCs has become an issue. This problem attracts researchers to optimize the workload allocation process with many objectives in mind such as makespan minimization, load balancing, cost reduction and energy enhancement, as well as network bandwidth improvement.

Banerjee *et al*. (2015), proposed a new greedy cloudlet allocation method to improve the makespan of VMs through load balancing. The proposed model improves VMs and hosts load balancing with QoS intention through minimizing the VMs and hosts Makespan as well as the cloudlets completion time. The model achieved a load balancing through distributing the predictable load to VMs based on their capacity to enhance system utilization. Dong *et al*. (2015), the task mapping was formulated as an integer programming optimization problem which is solved using a greedy algorithm. The proposed Most-Efficient-Server-First (MESF) scheduling scheme schedules tasks to most energy efficient servers to minimize DC energy consumption. The proposed model considers the average

queuing delay of tasks build up on some of the servers. However, simulation results showed that MESF operates poorly in case of independent and identical distributed task arrivals that follow an exponential distribution.

The Conductance cloudlets allocation policy proposed by Chatterjee *et al*. (2014), considering VMs capacity as a pipe by finding the conductance of each VM through the ratio of its processing speed to the sum of the processing speed of all available VMs. The proposed algorithm did not take into account the cloudlets length, which leads to VMs load balancing problem. Huai *et al*. (2013) studied the task scheduling problem in heterogeneous servers' environment to reduce the energy consumption. The authors proposed the Benefit-driven Scheduling (BS) method that maps tasks to the most suitable server type. For homogeneous systems, the paper proposed two allocation task scheduling using two different heuristic algorithms named Power Best First (PBF) and Load Balancing (LB). The proposed methods applied and tested on a Dynamic Voltage and Frequency Scaling (DVFS) homogeneous and heterogeneous server-based environment to study the effect of suitable working frequency in achieving power saving. The results showed around 13% power saving.

All the above works targeted the minimization of energy consumption and improving the QoS. However, they do not consider the problem of transmission delay and cost in selecting the most suitable computing resources for workload execution.

Kliazovich *et al*. (2013) investigated the role of the network fabric and proposed an energy efficient task scheduler with traffic load balancing, the e-STAB scheduler. The proposed method consolidated jobs to a minimum number of activated servers in order to minimize network congestion and delay. Their method studied the problem of network delay and energy efficiency inside one DC. Liu *et al*. (2013) proposed an energy efficient, profit and cost aware task scheduling and resource allocation in a multielectricity-market environment to maximize the net profit of cloud provider. The model incorporates the multi-electricity market, SLA and net profit as a multi-objective task-scheduling framework. The problem was formulated as a constrained optimization problem. The results showed the proposed method improved the net profit due to energy efficient using of computing resources.

This paper studies a new approach targeting data-intensive workload allocation in geographically distributed DCs. It incorporates the delay incurred due to the network link capacity in mapping the workload which leads to further reduction in the DC energy consumption. Also, the paper proposes a data location-aware algorithm to the task allocation problem that takes into consideration the communication costs between the distributed storage servers and computes servers when allocating the workload tasks/cloudlets to the Virtual Machines (VMs). Simulations on real workload traces show that the proposed work minimized the workload Makespan while reduced the DC energy consumption compared with other approaches.

Cloud System Model

This section defines the problem under investigation and describes the cloud system architecture used in this study. Our model targets geodistributed cloud environment for running dataintensive applications.

Some Definitions

Cloud-computing environment is a parallel and distributed system working at the same time to satisfy the users' needs (Buyya *et al*., 2009). It is the delivery of on-demand computing resources, everything from applications to data centers infrastructure, over the internet on a payper-use basis. A cloud broker is a third party that acts as an intermediary between the customers of cloud services and the service provider. Most of the cloud providers have DCs distributed over different sites interconnected via Wide Area Network (WAN). The mapping process of the user's workloads to the cloud computing resources (i.e. the VMs running on the data center servers) is called task allocation.

Target Application

This paper studies data-intensive applications where large data transfer times compared with their processing times characterizes them. This type of application may arise in for example distributed database query processing. Usually, the application data files reside on one or more servers of data centers and the computing phase might be done at another data center. Therefore, data are required to move from the storage site to the computing site to process user's query request and deliver information quickly and efficiently.

It is clear that the overhead of data transfer can dramatically degrade the performance of the application especially if the network traffic is not optimized. Therefore, efficient task allocation methods that move data to process efficiently are mandatory to enhance the overall completion time of data-intensive workloads.

Co-locating data and computation on the same DC to serve data-intensive applications would evidently lead to ideal performance. Nevertheless, this is not always possible. Another motivating scenario when a private and secured data of a company located on their local storage nodes. However, the company has limited computational resources and facing a deadline constraint. In this case, the company may leverage its resources by moving from their private computation resources to public cloud so that they can meet their deadline. In this particular scenario, an application provider aims are to serve the users' request with a good quality of service and within a deadline time determined in advance. Assume that requesting the service is through a given

broker of the service provider. The broker responsibility is to accept the users' requests, creating a task for each request and then allocates the tasks to the computing resources or virtual machines run specifically for this application. However, moving data to distant computational resources might become a bottleneck due to the data size and network bandwidth. Moreover, addressing computation separately from data movement would lead to performance degradation and SLA violations.

Specifically, this paper targets to schedule deadline constrained data-intensive workload applications. The user primarily transfers the data files to the local storage infrastructure of the cloud. The workload is consists of a set of parallel tasks; each one can run separately on a computing node. Additionally, each task is associated with a data file, residing within the broker storage node. Finally, all the workload tasks must finish within the deadline constraint.

System Model

Geographically distributed cloud environments, such as Amazon and Google, provide data storage clouds, such as Amazon S3, for data storage and compute clouds, such as Amazon EC2, as services. However, storing the application's data in different storage nodes/servers than the running application's processes will definitely incur high data transmission due to the cost of moving data back and forth over network interconnection links (Piao and Yan, 2010). Our objective in this study is to minimize the submitted workload makespan to improve the users QoS while reducing the energy consumption of the compute resources.

Throughout this paper, a task will be considered as the smallest unit work in a user's submitted data-intensive application. An application may consist of one or more tasks. The set of parallel tasks referred as a BoT. Each task has a length measured in Millions of Instructions (MI). Although it is difficult to predict the number of instructions executed by each task, however, in the literature different smart models are constructed for this purpose (Kumar and Singh, 2018; Ha *et al*., 2018; Toosi *et al*., 2018; Al-Dulaimy *et al*., 2016) such as the prediction model developed by the authors in (Ha *et al*., 2018) to describe the requirements of tasks and to estimate the cost of running that task on an arbitrary resource using baseline measurements from a reference machine.

Tasks are allocated to a set of running VMs which is initialized in advance. This is due to the overhead of setting up and creating a new VM. The pre-allocation of the VMs, as done in Azure cloud environment (https://docs.microsoft.com/enus/azure/guidance/guidance-compute-multipledatacenters; Mazumdar *et al*., 2016), achieves high availability across regions.

As depicted in Fig. 1, the cloud system model considered in this study has a set of DCs distributed across different geographical sites. A broker accepts the requests of users and places their data files on some Storage Node (SN) on a different DC. We assume the links between the SN and the VMs on the DCs are set up to move the user's data files to and from the DCs hosting the VMs.

Problem Formulation

This section presents our mathematical formulation of the task allocation problem of data-intensive workloads as a nonlinear optimization problem. Our main objective is to minimize the workload makespan by reducing the network delay due to the transfer of tasks' data transmission and to minimize the power consumption of servers which is expected to reduce the DCs energy consumption, as is shown in the experimental results section.

Model Assumptions

Before proceeding with the problem formulation, the following assumptions are taken into consideration:

- The workload consists of a set of independent tasks (called bag-of-tasks or BoT). Therefore, there are no dependencies between the tasks.
- Each task has a prior known length measured in Millions of Instructions (MI).
- The VMs are already initialized and running on servers of specified DCs.
- The BoT workload is submitted to a broker together with the task's associated data files associated which are stored in a specified storage node/server.
- There are dedicated network links between the storage node and the distributed data centers where the VMs reside on.
- The available VMs have different computing power (MIPS)
- The space-shared policy is a VMM allocation policy that allocates one or more Pe to a VM and doesn't allow sharing of PEs. If there is no free Pes to the VM, allocation fails. Free PEs are not allocated to VMs.
- A task is allocated and executed in only one VM (computing resource).

Model Formulation

Table 1 defines the different parameters used in our formulation. Given a set of n tasks, $T = \{t_1, t_2, \ldots, t_n\}$, a set of m files, $F=\{f_1, f_2, \ldots, f_m\}$, associated with the tasks T such that $n = m$, a set of data centers $D = \{dc_1, dc_2, \ldots, dc_s\}$ distributed in different sites such that each site might host more than one DC. The following equations define how the computing and transfer times are estimated and used in calculating the makespan of the total workload.

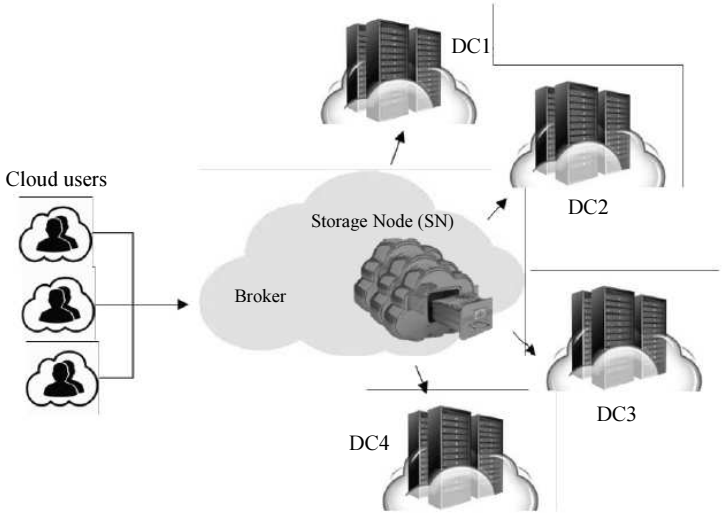

Fig. 1: Cloud provider environment

Table 1: Symbols used in LAEE problem

Notation	Description
T	Independent set of tasks in a workload called a bag-of-tasks (BoT)
t_i	A single task (cloudlet) submitted by a user, $t_i \in T$
F	The set of input and output files associated with tasks T
f_i	A data file associated with a task j, $f \in F$
D	A set of geographically distributed data centers
dc_d	A data center at a given location s.t. $dc_d \in D$
R	Set of k compute resources, i.e., VMs
r_j^d	Compute resource j on dc_d *s. t.* $r_j \in$ R and $dc_d \in D$
$ExecTime(t_i, r_j^d)$	Execution time of task t_i on compute resource r_j hosted in data center dc_d
$TranTime(t_i, r_j^d)$	Transfer time associated with task t_i to be executed on compute resource r_j on dc_d
$InputSize(f_i)$	the input file size associated with task t_i
$OutputSize(f_i)$	the output file size associated with task t_i
T_t	A set of tasks running on a compute resource r_j^d *s.t.* $T_t \subset T$
$Length(t_i)$	Length of a task t_i (measured in million instructions)
$mips(r_j^d)$	Available mips for compute resource r_j at dc_d
$pe(r_j^d)$	The number of processing elements assigned to compute resource r_j^d at d_{cd}
$Available\ (LinkBw(dc_d))$	available link bandwidth capacity between the broker (storage node) and data center d hosting resource r_j^d due to interaction of different file transfers on the same link
$delay(dc_d)$	The link delay time to transfer data to data center d hosting resource r_j^d
DeadlineT	The deadline time constraints for a BoT T given by the SLA between the provider and user

The computation time of task t_i on compute resource r_j^d is defined as (Banerjee *et al*., 2015):

$$ExecTime(t_i, r_j^d) = \frac{Length(t_i)}{mips(r_j^d) * pe(r_j^d)} \quad (1)$$

We assumed in (1) that a task can run on all the available cores or processing elements (Pes) owned by the VM (i.e., r_j).

The transfer time to move the input and output files associated with task t_i to the compute resource r_j on the data center dc_d can be estimated as:

$$TranTime(t_i, r_j^d) = \frac{InputSize(f_i) + OutputSize(f_i)}{Available\ (LinkBw(dc_d))} + 2 * delay(dc_d) \quad (2)$$

In Equation (2), *InputSize* (f_i) and *OutputSize* (f_i) are the input file and output file sizes associated with task t_i, the Available(*LinkBw*(dc_d)) is the available link bandwidth between the Storage Node (SN), where files reside in and the compute resource r_j^d located on data center dc_d due to interaction of different file transfers on the same link. The *delay*(dc_d) is the delay time of the network link used to move the input and output data files associated with task t_i on compute resource r_j^d.

The total execution time of a task t_i, including both computing and communication time, (i.e., the turnaround time) to be processed on compute resource r_j is given as:

$$CompletionTime(t_i, r_j^d) = \begin{cases} TranTime(t_i, r_j^d) + ExecTime(t_i, r_j^d) \\ if\ there\ are\ available\ resources \in r_j^d \\ CompletionTime(t_i, r_j^d) + TranTime(t_i, r_j^d) \\ + ExecTime(t_i, r_j^d),\ otherwise \end{cases} \quad (3)$$

In Equation (3), we assumed that the VMScheduler (In CloudSim, VmScheduler is an abstract class that represents the policy used by a Virtual Machine Manager (VMM) to share processing power among VMs running in a host) used in our CloudSim simulations would follow the space-shared policy to execute the BoT workload in case of unavailable resources in r_j^d. Noting that *CompletionTime* (t_x, r_j^d) is the completion time of the recently ended task, where x is the index of the recent ended task.

Given the completion time of a task in Equation (3) above, the makespan for executing the set of t tasks, $T_t = \{t_1, t_2, \ldots, t_t\}$, on a virtual machine r_j^d *s. t.* $T_t \subset T$, will be the completion time of the last executed task t_t, as shown in Equation (4) and Fig. 2:

$$Makespan(T_t, r_j^d) = CompletionTime(t_t, r_j^d) \quad (4)$$

Figure 2 shows a possible situation when t tasks are assigned to one virtual machine r_j^d.

Now, our objective is to allocate the BoT workload consisting of tasks T to the set of available computing resources R running on the set of data centers D so that the total makespan time is minimum. Equations (5), (6) and (7) define the formulated optimization problem:

$$\text{minimize}\left(\max_{\substack{1 \le j \le k \\ 1 \le d \le S \\ T_t \subset T}}\right)\left(Makespan\left(T_t, r_j^d\right)\right) \quad (5)$$

Such that:

$$Makespan(T, D) \le Deadline_T \quad (6)$$

The constraint in Equation (6) ensures that completion time of the whole workload should not exceed the deadline time constraints given by the SLA agreement between the provider and the workload users. Noting that the violation of the deadline constraint can lead to undesired consequences for the user as well as the cloud provider.

Example

This example demonstrates how our proposed model can be used to allocate a small Bot workload with four tasks {C1, C2, C3, C4} so that the makespan is the minimum between all possible allocations. We assume two different geographical distributed environments, {dc1, dc2}. Each data center has one compute resource (VM). Figure 2 shows a sketch of the model using the numbers mentioned above. The values of the different parameters associated with the workload tasks and the cloud system are given in Table 2.

As Fig. 3 shows, a broker needs to find the best virtual machines in the available data centers to allocate the incoming workload so that total makespan of the workload is minimum and does not exceed the deadline time constraints determined in the SLA agreement between the provider and workload users.

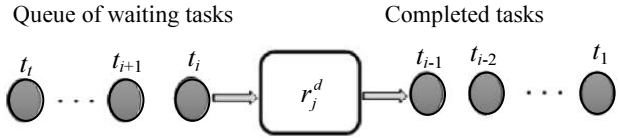

Fig. 2: Scheduling t tasks to r_j^d

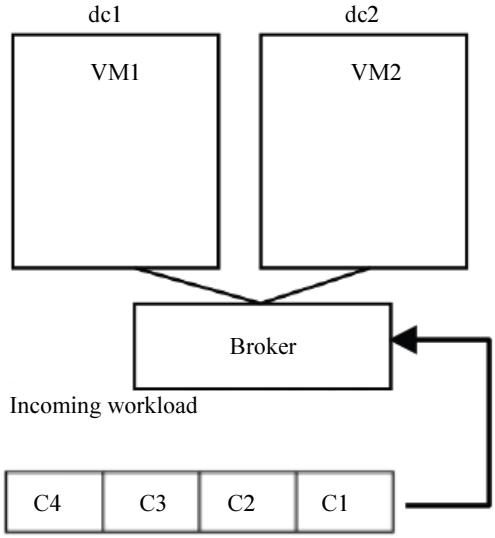

Fig. 3: Cloud service provider environment model

The objective is to find an optimal allocation to the workload {C1, C2, C3, C4} with a minimal makespan among all possible allocations. Table 3 shows the calculated makespan for all possible cloudlets allocation in this example (16 in this example). We show the estimated transfer and execution times according to Equations (1-3) for the scenario "the 4 cloudlets are allocated to VM1 on dc1". Since there is one processing core, the first cloudlet arrives will be assigned the physical core while the rest of cloudlets will be waiting in a queue as shown in Fig. 4. Using the formulation discussed in section 4 the output will be as follows:

$$TranTime(C1_{VM1}) = \left(3*10^9 + 3*10^9\right) / \left(10*10^9\right) + 2*50*10^{-3} = 0.7$$

$$ExecTime(C1_{VM1}) = 1000 / \left(1000*1\right) = 1$$

$$CompletionTime(C1_{VM1}) = 0.7 + 1 = 1.7$$

$$CompletionTime(C2_{VM1}) = 1.7 + 0.7 + \left(2000 / 1000*1\right) = 4.4$$

$$CompletionTime(C3_{VM1}) = 3.4 + 0.7 + \left(3000 / 1000*1\right) = 8.1$$

$$CompletionTime(C4_{VM1}) = 8.1 + 0.7 + \left(4000 / 1000*1\right) = 12.8 \quad (7)$$

Therefore, the makespan of the workload consisting of the 4 cloudlets is equal to 12.8 with Makespan of 11.8.

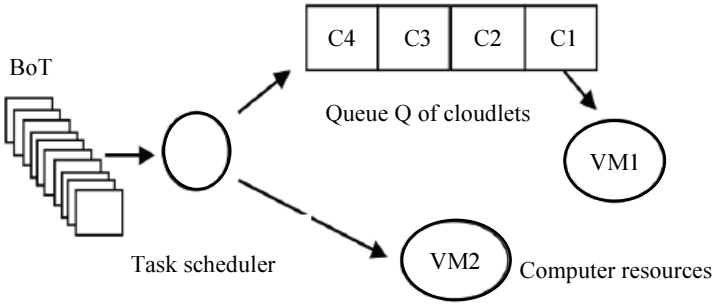

Fig. 4: Data center task scheduler model

Table 2: The specifications of the cloud model discussed in the above example

Entity type	Parameter	Name	Value
Cloudlets	Cloudlet length (MI)	C1	1000
		C2	2000
		C3	3000
		C4	4000
	Input/Output file size (GB)	C1,C2,C3,C4	3
VMs	CPU(MIPS)	VM1	1000
		VM2	2000
	Number of cores (PE)	VM1, VM2	1
Data centre	Delay (milliseconds) (between broker and DC)	Dc1	50
		Dc2	100
	Bandwidth (Gbp/s)	Dc1 and Dc2	10

Table 3: Different allocations of the given BoT workload with the Makespan of each

Scenario number	Cloudlets/VMs	Makespan
1	{C1,C2,C3,C4;VM1}	12.8
2	{C1;VM1}, {C2,C3,C4;VM2}	10.4
3	**{C1,C2:VM1},{C3,C4:VM2}**	**5.1**
4	{C1,C2,C3;VM1},{C4;VM2}	8.1
5	{C2;VM1}, {C1,C3,C4;VM2}	6.4
6	{C2,C3;VM1},{C1,C4;VM2}	6.4
7	{C2,C3,C4;VM1},{C1;VM2}	11.1
8	{C3;VM1}, {C1,C2,C4;VM2}	5.9
9	{C3,C4;VM1},{C1,C2;VM2}	8.4
10	{C1,C3,C4;VM1},{C2;VM2}	10.1
11	{C4;VM1}, {C1,C2,C3;VM2}	5.4
12	{C1,C4;VM1},{C2,C3;VM2}	6.4
13	{C1,C2,C4;VM1},{C3;VM2}	9.1
14	{C1,C2,C3,C4;VM2}	8.2
15	{C2,C4;VM1},{C1,C3;VM2}	7.4
16	{C1,C3;VM1},{C2,C4;VM2}	5.4

The optimal solution in the example refers to Scenario 3 since the makespan will be 5.1, the 6 minimal among all possible 16 allocations in Table 3. In general, if we need to allocate a workload of t tasks to r compute resources we need to explore a total of $\sum_{i=0}^{t} \frac{t!}{i!(t-i)!}$ possible allocations since the task allocation problem is a Nphard problem.

DVFS and Power Consumption Model

Dynamic Voltage and Frequency Scaling (DVFS) is an effective technique to reduce servers' power consumption through scaling the CPUs frequency proportionally to their loads (Maiti and Sudeep, 2017). Although DVFS can significantly achieve power efficiency in computing resources, it also reduces the computing performance (Huai *et al*., 2013). However, since the paper targets to schedule non compute-intensive workload application, DVFS technique will be a useful approach to achieve power efficiency (Maiti and Sudeep, 2017). Consequently, this paper, incorporate DVFS mechanism with the proposed model (presented in section 4.1) targeting to minimize the energy wasting which becomes one of the key challenges that affect the cloud sustainability. DVFS is supported by most modern CPUs to scale down its frequency and voltage when it is not fully utilized (Huai *et al*., 2013). The quadratic relation between the frequency adjustment and the CPU dynamic power consumption is shown in the following (Huai *et al*., 2013):

$$P_{CPU-Dynamic} = ACV^2 f \tag{8}$$

where, A, C, V and f are the switching activity, the physical capacitance, the supply voltage and the clock frequency respectively. Voltage V can be expressed as a linear function of frequency such that, $V = af$ and a is constant. Therefore, Equation (9) express the new form of dynamic CPU power:

$$P_{CPU-Dynamic} = \beta f^3 \tag{9}$$

Developing an energy-aware task allocation algorithm requires measuring the dynamic power consumption resulted from running the tasks on the compute resources (VMs). To derive a power consumption model, real time server power consumption monitoring is needed. However, this is out of the scope of this paper. Instead, we used the cubic frequency-power approximation model (Huai *et al*., 2013). In this study, we only consider the power consumed by CPU; therefore, Equation (10) is used to find the total power consumption of a server:

$$Power = P_{CPU-Static} + P_{CPU-Dynamic} \tag{10}$$

where, $P_{CPU\text{-}Static}$ is the static power consumption denoted as γ and $P_{CPU\text{-}Dynamic}$ calculated as in Equation (9). Therefore, using Equations (9 and 10), the cubic power consumption model that computes the total power consumption of a server is as follows:

$$Power = \gamma + \beta f^3 \tag{11}$$

Equation (10) shows clearly that frequency f is the only variable value that effects the servers' power consumption where γ and β are constants and varies among different servers.

Consequently, the power consumption of a server h_k holding number of computer resources r_j^d on data centre dc_d is denoted as $Power(h_{k,d})$, nothing that each host can hold more than one r_j. Let's consider $Power_{TD}$ as the total power consumed by R computer resources to process T request of tasks. Therefore, the target is to minimize the total value of *Power* as follows:

$$\text{minimize} \sum_{d=1}^{|D|} \sum_{j=1}^{|R|} Power(r_j^d) \tag{12}$$

where, $Power\, r_j^d$ is the power consumption of the r_j on dc_d calculated using Equation (8).

LAEE Model

The objective of the energy efficient LAEE workload allocation model is to minimize the workload makespan and attain maximum energy efficiency when mapping a set of tasks T to the set of available resources R without violation of the Deadline constraint DTT given by the SLA between the provider and cloud user. This could be achieved through considering the objectives described in Sections 4.2 and 4.4. Therefore, the objective of the proposed model is as follows:

$$\text{minimize}\,(Makespan(T,D), Power_{TD}) \tag{13}$$

subject to Equation 6 constraint that ensures that the completion time for a BoT T on a set of available Resources R should not exceed the deadline time constraints DTT given by the SLA between the provider and user:

$$Makespan(T,D) \leq Deadline_T$$

The LAEE Workload-Allocation Algorithm

Genetic Algorithms (GA) are adaptive heuristic random optimization algorithm that works via the process of natural selection and evolution (Golberg, 1989). In this section, we propose the Location-Aware and Energy-Efficient (LAEE) workload allocation genetic algorithm to solve the formulated non-linear optimization problem given in Equations (5-13). The task allocation problem we target in this study is NP-hard (Nemhauser and Wolsey, 1988). Therefore, our proposed algorithm will try to find a near-optimal solution heuristically based on genetic programming.

Given a BoT workload n tasks $T = \{t_1, t_2, \ldots, t_n\}$, a set of associated m files $F = \{f_1, f_2, ..., f_m\}$, a set of data centers $D = \{dc_1, dc_2, \ldots, dc_s\}$ located in different geographic locations and a set of compute resources $R = \{r_1, r_2, \ldots, r_k\}$ hosted on the data centers D. The building blocks o the genetic LAEE algorithm are described as follows.

Encoding

This is the representation step of the optimization problem in terms of genetic terminology. A chromosome in the GA consists of n genes corresponding to the n tasks of the workload T. That is, each gene represents the mapping of a task t_i to a specific Virtual Machine (VM), or compute resource r_j. The value of a gene is a positive integer representing the VM number where the task is allocated. Table 4 shows an example of mapping 15 tasks to 4 VMs and its corresponding chromosome consisting of 15 genes.

Initial Population

A population is a set of solutions to the original optimization problem, i.e., a set of possible allocations of T to R. An initial population is a set of chromosomes (solutions) that are randomly created as a starting step towards finding an optimized solution. Each chromosome in the population represents a candidate solution to the problem and it is called an individual. The fitness function is calculated to the individual. Then, a number of fittest individuals are selected to mate and produce a next enhanced population (or generation). To generate efficient and robust genetics search space diversity is taken into account through generating the initial population that gives the genetic better possibility to find a good and near-optimal solution (Yin *et al*., 2017).

Fitness Function

Selecting a suitable fitness function is significant to design a successful Genetic Algorithm. Since the goal is to minimize the makespan of the workload T, the fitness function is chosen to be the objective function of the formulated optimization problem given in Equation (5).

Selection

The selection method dictates how to choose the individuals in a population and use them to produce a new generation so that a better solution is obtained. There are various strategies to select the best individual such as Boltzmann strategy, rank based selection, roulette wheel and tournament selection (Golberg, 1989). This paper uses the roulette wheel based on a rank given to each individual according to its fitness value.

Crossover

This is the method of selecting two parents (individuals) to produce next-generation individuals. We used the mostly used random point crossover technique since it helps to exchange VMs assignment between corresponding tasks (Yin *et al*., 2017). Figure 5 shows a simple example of randomly selected crossover point on both selected parents and the newly generated individuals (children) that outline the newly produced generation.

Mutation

Mutation maintains genetic diversity in the subsequent generations. It avoids generating uniform populations. The mutation operator is used to modify the genes of a randomly selected chromosome according to a mutation probability.

LAEE-GA

In this study, a modified version of GA is proposed to solve the LAEE optimization problem (Equations 5-13). Based on the basic operations discussed above, the LAEE genetic algorithm (Algorithm 1) starts by creating initial random population (line 2) using encoded binary (0, 1). The main important part of the proposed algorithm is fitness function evaluation that reflects the main objective of the proposed model in minimizing the workload makespan (line 3). Line 3 evaluates the fitness function of each individual using Equation 5. Then genetic operators applied through crossover and mutation operations to the selected parents (line 4-10). After each iteration, new population is created using fittest individuals. After a number of iterations, the algorithm retrieves the individual with the highest fitness from the last population as a near-optimal solution to the proposed problem line (9).

The LAEE genetic allocation algorithm designed to solve the problem of data-intensive workload allocation and to attain a trade-off between energy efficiency and QoS. Most of the modern computers integrated with an effective dynamic DVFS mechanism (Maiti and Sudeep, 2017). Accordingly, Algorithm 1 executed in cloud DVFS environment, i.e. we assume that the power consumption of active servers will scales linearly with its CPU utilization. Reducing CPU frequency minimizes the CPU power consumption (as Equation 11 depicts). However, this could not lead to energy saving since reducing frequency implies that more time will be taken to handle the given workload (Huai *et al*., 2013). Nevertheless, the LAEE allocation algorithm that targets to minimize the workload makespan shows its contribution on energy saving (as shown in Section 6.3.2) compared to other competitive algorithms that employ a DVFS mechanism.

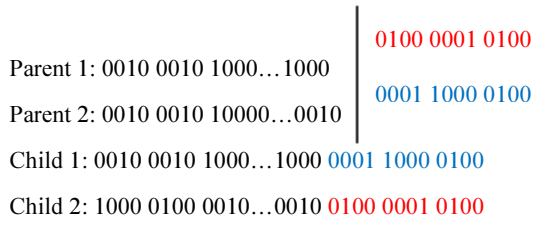

Fig. 5: Random crossover point

Table 4: Tasks as a chromosome represent the allocation of the tasks to the available VMs

Task#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
VM#	3	4	1	2	4	1	3	3	1	4	2	1	3	2	1
Binary represent of genes	1	10	1000	100	1	1000	10	10	1000	1	100	1000	10	100	1000

Algorithm 1: LAEE

Input: $T = \{t_1, t_2, \ldots, t_n\}$, a set of workload tasks
$R = \{r_1, r_2 \ldots r_k\}$, a set of distributed compute resources i.e. VMs.
$D = \{dc_1, dc_2, \ldots, dc_s\}$, a set of s data centers.
Population, generation size, $Deadline_T$, maximum number of generations.
Output: allocating tasks T to R.
Processing:
1: Begin
2: Initialize population (Select first generation from a pool of genes that can randomly allocate the set of tasks T to the compute resources R)
3: Calculate the fitness value for each individual in the initial population
4. Find the top two fittest individuals and consider them elite and pass them to next generation without any changes.
5. Do
    5.1. Using random Roulette Wheel method to select two parents
    5.2. Perform crossover between the two parents
    5.3. Pass the individuals to the next generation
6. Until the new generation size = the initial population size
7. Replace current generation with the newly created generation.
8. Apply mutation to the genes with some probability and place the resulting chromosomes in the new population
9. Go to step 3 until the maximum number of generations is reached or if the value of the calculated fitness function is less than or equal to $Deadline_T$ (Equation (6)).
10. End

Performance Evaluation

This section validates the performance of the proposed LAEE algorithm through extensive simulation experiments conducted using the CloudSim 3.0.3 simulator. The performance results of the proposed LAEE algorithm are analyzed and compared to the benchmark task allocation algorithms, namely, Round Robin (RR) and Shortest Job First (SJF, it sort the BoT in increasing order to task's length that measured in MI). Also, we compared our algorithm with another geneticbased task allocation algorithm (Kumar *et al*., 2015), which we will name it (GGA) for ease of reference. Since the objective is to enhance the QoS of the workload, four time-based metrics, namely, workload makespan, VM makespan, host makespan, task execution time are measured in the comparison study. We also show how our allocation algorithm succeeded to reduce the total energy consumption when employing the DVFS strategy to adjust the hosts' frequencies based on their CPU utilization.

Simulation Setup

We conducted experiments on Intel(R) core(TM) i7 Processor running at 3.4GHz with Windows 7 Operating system and using NetBeans IDE 8.0.2 and JDK 1.8. We generated a number of simulations by varying the number of hosts or Physical Machines (PMs) and their specifications, the number of compute resources (VMs) and their configurations, the number of geographically distributed data centers and the network's links delay costs to the Storage Node (SN) where the data files initially reside and the requirements of the workload tasks.

Our simulations tackle two different scenarios. The first uses synthetic traces, which randomly models the cloud computing environment to measure the effectiveness of the proposed method using the timebased metrics. The second scenario uses the benchmark Planetlab workload traces (PLT, 2016). The specifications of the hosts used to measure the power consumption due to using our proposed algorithm are HP ProLiant ML110 G4 (1 × [Xeon 3040 1860 MHz, 2 cores], 4GB) and HP ProLiant ML110 G5 (1 × [Xeon 3075 2660 MHz, 2 cores], 4GB) (SPEC, 2016).

We extended the CloudSim environment to implement three different matrices that represent the links bandwidths, the links delays and the computed cloudlets (tasks) execution times. The bandwidth matrix represents the bandwidth link capacity between the SN data center (broker side) and the data centers hosting the VMs. The links' delay matrix stores the values of the average communication delay measured between the SN data center and the different data centers hosting the VMs (i.e. as if we are modeling Google cloud computing environment with distributed data centers in America, Asia and Europe). Although the distance is not an ideal estimator for network latency, it is sufficient to determine the relative rank in latency from end-user to data centers as indicated in (Fan *et al*., 2016). Moreover, we use the WAN Latency Estimator (http://wintelguy.com/wanlat.html) to estimate the network latency in milliseconds used in our simulations. The estimated execution cost of each the workload tasks (cloudlets) allocated to a specific VM on a DC is calculated and stored in a temporary matrix based on our modeling in Equations (1-3).

Table 5 shows the values of the different parameters used in the genetic operations of our proposed metaheuristic algorithm, LAEE.

Experimental Results

Scenario 1

This scenario uses synthetic data that randomly model the cloud-computing environment. Table 6 shows the ranges of values for the parameters used to model the cloudlets/tasks, the data centers, the hosts and the virtual machines (compute resources).

We compared the performance of our workload allocation algorithm LAEE with RR, SJF and GGA algorithms. Note that, the plotted results of LAEE and GGA algorithms are the average output of 20 independent executions.

Workload Makespan

Figure 6 shows the workload Makespan compared to the three task-allocation algorithms RR, SJF and GGA. As the result shows, the improvement rate of the BoT Makespan ranges between 15% over GGA method and about 28% over RR and SJF policies. This improvement has a direct effect on the cloud QoS since it reflects the cloud user requested services completion time.

VM Makespan

Figure 7 and 8 shows the makespan of the compute resources (VMs) and it compares with the other algorithms. Actually, this performance metric is an indicator of the success of an allocation algorithm to distribute the workload tasks on the available VMs so that the utilization of the VM is reduced. Figure 7 shows that LAEE algorithm has approximately a uniform VM Makespan among all available VMs compared to RR and SJF, which reflects the distribution and load balancing of the workload on available VMs. Also, Fig. 8 shows how our proposed LAEE algorithm proves its efficiency in VM Makespan reduction over a randomly selected 4 VMs.

Host Makespan

Figure 9 and 10 shows the measurement of the host makespan among the four algorithms. Figure 9 shows the rate of improvement in the host's Makespan for randomly selected hosts. Figure 10 reveals that there is an 8% improvement on host makespan compared to GGA and a 25% improvement compared to RR and SJF. The achieved results reflect the importance of the proposed method on load balancing. This improvement has a direct outcome on the cloud QoS and a great effect on improving cloud energy consumption.

Table 5: Genetic parameters settings

Parameter	Value
Population size	100.00
Number of generations	100.00
Crossover rate	0.80
Mutation rate	0.15

Table 6: Cloudsim parameter settings

Entity type	Parameter	Value
Cloudlet	Cloudlet length (MI)	200-4000
	Input/Output file size (MB)	3000-8000
	Number of cloudlets	500
Data center	Number of distributed Data Centers (DC)	2
	Type of data centers	Heterogeneous
	Link delay (milliseconds) between SN (broker) and DCs	10-100
	Bandwidth (Gbp/s)	1-10
Host	Number of hosts	8
	dc1 hosts' names	h0, h1, h2, h3
	dc2 hosts' names	h00, h01, h02, h03
	Number of Cores	1-4
	MIPS/CPU	2000-4000
	RAM (GB)	16
VM	Total number of VMs	24
	CPU (MIPS)	100-700
	Number of cores per VM	1

Table 7: Cloudsim parameter settings

Cloud resources	Small	Medium	Large
Number of cloudlets	500	1000	1500
Number of VMs	30	100	160
Number of distributed DCs	5	7	10

Task Execution Time

Figure 11 represent the task execution time improvements on a randomly selected bunches of tasks/cloudlets from a 500 cloudlets. However, Fig. 12 reflects the 500 cloudlets task execution time enhancement compared to RR and SJF benchmark methods.

Workload Size

As the number of cloudlets and compute resources increase the improvement rate of LAEE algorithm increases, this is expected from the nature of meta-heuristic approaches. Consequently, a genetic-based algorithm, like ours, is expected to reach a near-optimal satisfactory solution to the optimization problem provided the search space is large (Chu and Beasley, 1997). Figure 13 shows the LAEE algorithm improvement in getting a minimal makespan to the BoT workload when the workload size (number of tasks) is between 500 and 1500 while the number of VMs ranged between 30 and 160 and the data centers distributed among 5 to 10 geographical different locations. Table 7 shows the ranges of values for the parameters used in this experiment, where the sizes are categorized into small, medium and large classes.

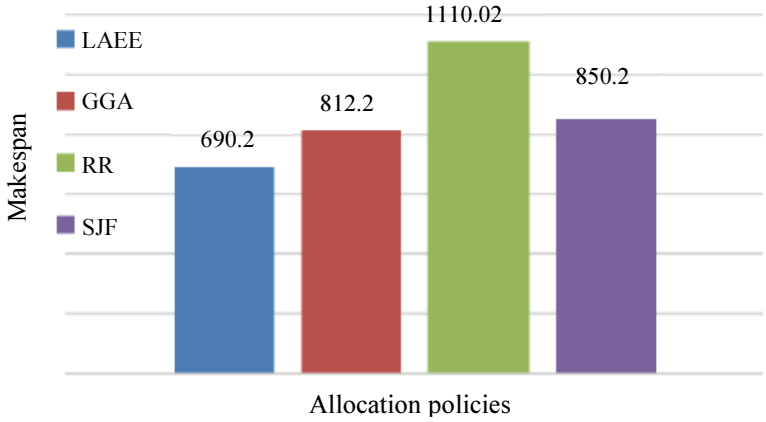

Fig. 6: Comparison graph of BoT Makespan among RR, GGA and SJF versus proposed LAEE algorithm

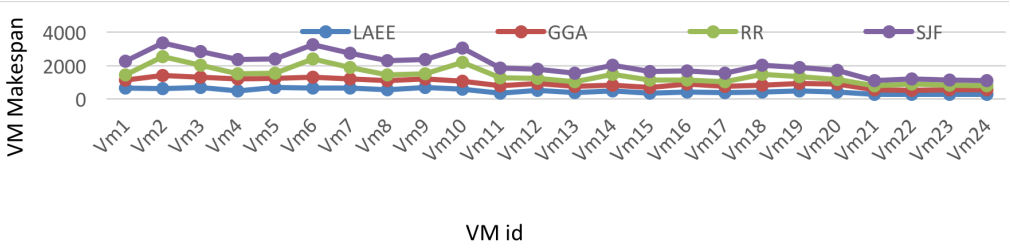

Fig. 7: Comparison graph of VM Makespan among RR, GGA, and SJF versus proposed LAEE algorithm

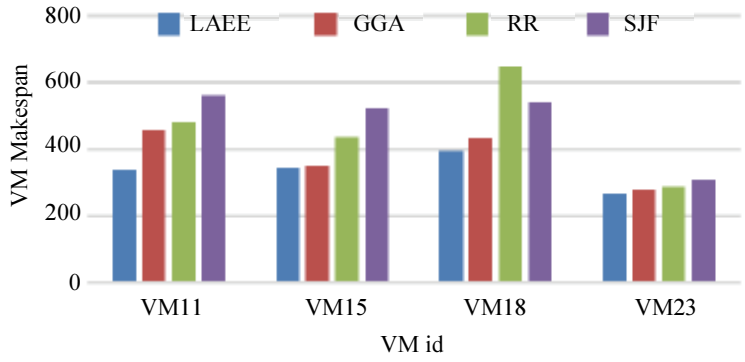

Fig. 8: Comparison graph of VM Makespan for randomly selected 4 VMs among RR, GGA and SJF versus proposed LAEE algorithm

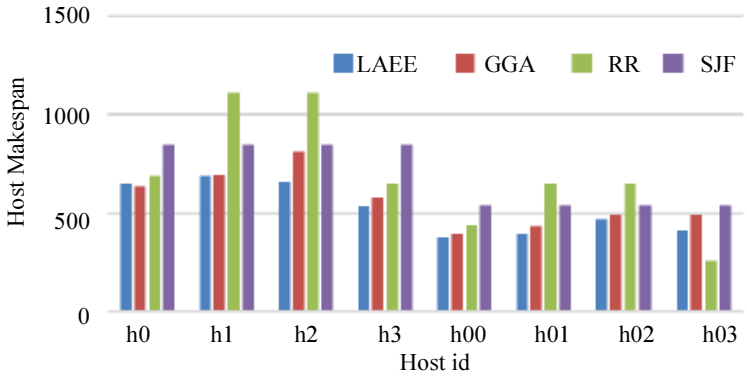

Fig. 9: Comparison graph of Host Makespan among RR, GGA and SJF versus proposed LAEE algorithm

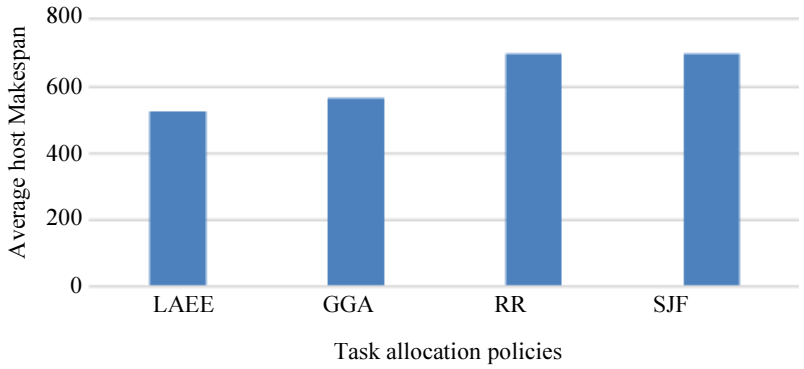

Fig. 10: Comparison graph of average host Makespan among RR, GGA and SJF versus proposed LAEE algorithm

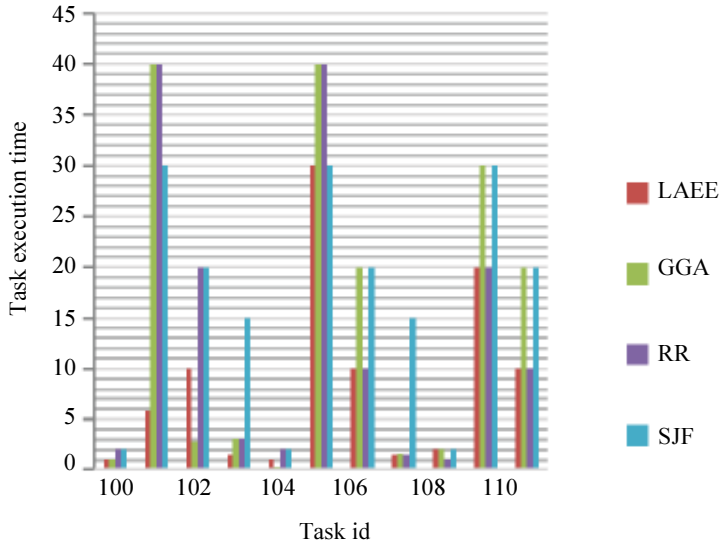

Fig. 11: Comparison graph of Task Execution Time for randomly selected 10 cloudlets among RR, GGA and SJF versus proposed LAEE algorithm

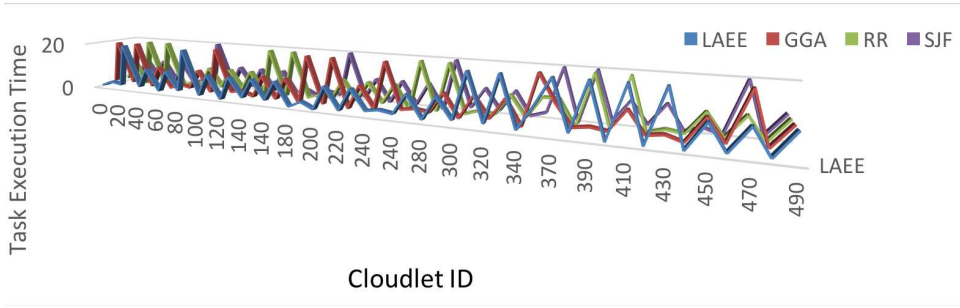

Fig. 12: Comparison graph of Task Execution Time for 500 coudlets among RR and SJF versus the proposed LAEE algorithm

Deadline and SLA Violation

SLA violation due to deadline constraint can reduce user satisfaction and degrade cloud providers' QoS. This is beside the penalty ratio that paid by cloud provider for consumers' compensations if the given deadline is missed. Literature and researchers reveal that poor cloud experience and delay for each one-second result in 16% degradation in customer satisfaction and more than 22% drop in cloud services sales (Bilal *et al*., 2018; Cheng *et al*., 2016).

The previous set of experiments and results show the importance of LAEE algorithm in meetings its target through minimizing the whole workload makespan to meet users' deadline constraint. Using different

configurations and cloud resources (as shown in Table 7), Fig. 14 compares the actual workload makespan with users' time constraint. It clearly displays that LAEE algorithm guarantees least number of SLA violations. For the first two sets of the experiment (Small and Medium), the SLA violation due to deadline constraint using LAEE algorithm is less 60% compared to other competing algorithms. However, in a Large set experiment, the SLA violation using LAEE algorithm is less than 0.5% compared to actual time constraint and less 80% compared to other algorithms. This is due to the nature of meta-heuristics GA algorithms that operates in a high performance using large search space to find a near-optimal solution (Fong *et al*., 2018).

Scenario 2

This test scenario is mainly provided to show the effectiveness of our proposed algorithm in reducing the total power consumption of the provider's data centers. We employed the DVFS techniques incorporated in the CloudSim simulator to benefit from any low utilization of the hosts to reduce the working frequency which contributed in reducing the energy consumption. This secondary objective (reduce energy consumption) is helpful in producing green computing while it directly leverages the revenue of cloud providers.

The experiment runs real Planetlab workload traces (PLT, 2016). The selected workload is made up of 302,976 Cloudlets with different cloudlets lengths. There are 800 heterogeneous hosts varies between HP ProLiant ML110 G4 (1 x [Xeon 3040 1860 MHz, 2 cores], 4GB) and HP ProLiant ML110 G5 (1 x [Xeon 3075 2660 MHz, 2 cores], 4GB) to run 1052 heterogeneous VMs with Amazon specifications.

The aim of this experiment is to find the effect of energy efficient task allocation on other management methods. The DVFS technique (Maiti and Sudeep, 2017) is used to adjust the hosts' CPU frequency according to their CPU utilization from executing the allocated tasks to it. In CloudSim, the frequency dynamically adjusted based on the CPU utilization percentage (Calheiros *et al*., 2011).

Figure 15 shows the energy consumption improvements of the used data centers when incorporating the DVFS technique in the proposed LAEE task allocation algorithm, rather than just using non-power aware allocation algorithms with DVFS technique. As shown in Fig. 15, the average power saving improvement rate due to using our LAEE algorithm is about 8% over the using the GGA, RR and SJF allocation algorithms.

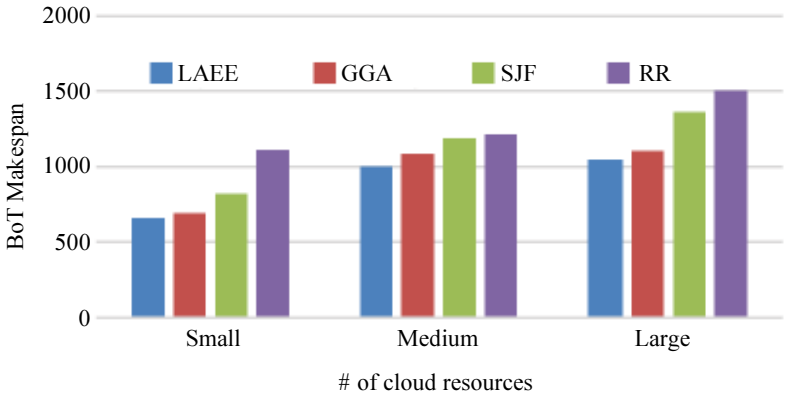

Fig. 13: BoT Makespan in different number of cloudlets and VMs

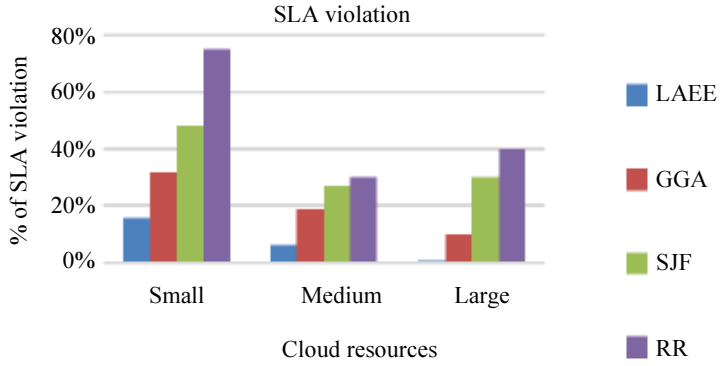

Fig. 14: BoT SLA deadline violation in different number of cloudlets and VMs

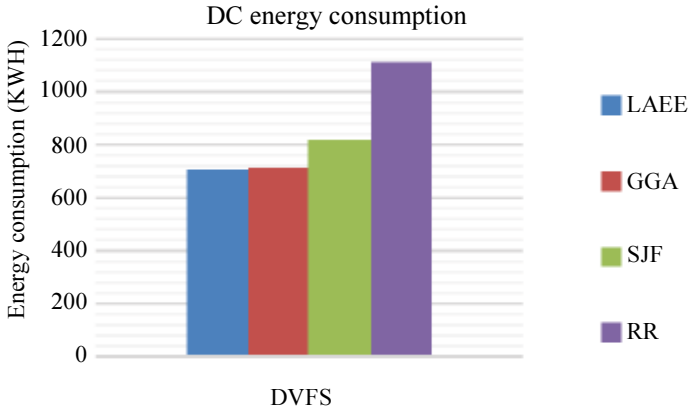

Fig. 15: Comparison graph of DC Energy Consumption among RR, GGA, and SJF versus proposed LAEE algorithm

It is worth to say that, although reducing CPU frequency minimizes the CPU power consumption (as Equation 11 depicts). However, this could not lead to energy saving since reducing frequency implies that more time will be taken to handle the given workload (Huai *et al*., 2013). Nevertheless, the LAEE allocation algorithm that targets to minimize the workload makespan shows its contribution on energy saving through effective load balancing (as shown in Section 6.3.1 – Fig. 7) that leads to makespan as well as energy minimization compared to other competitive algorithms that employ a DVFS mechanism.

Conclusion

This paper investigated the problem of dataintensive workload allocation in geo-distributed cloud environment. The problem formulated as a nonlinear programming optimization problem. Since this problem is known to be NP-hard, a meta-heuristic genetic algorithm is employed to find an optimized solution to the problem. The proposed algorithm considers both the task execution time as well as the cost of the data files transfer time from the storage location to the compute servers.

Extensive simulations are conducted using both synthetic and real traces on the known CloudSim cloud simulation package to prove the effectiveness of our proposed method. The results show the superiority of our algorithm over known allocation algorithms in minimizing the total makespan of a bag-of-tasks workload when executed on geo-distributed compute resources.

The DVFS technique is combined to our algorithm to show the effectiveness of our proposed algorithm in reducing the total energy consumption of the providers' data centers, which consequently contributed in maximizing the profit while keeping the users QoS improved.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgements

The authors wish to thanks Beirut Arab University for supporting this work.

Author's Contributions

Soha Rawas: Contributed to the LAEE concept. Wrote the initial draft and conducted the experiments.

Ahmed Zekri: Contributed to the LAEE concept. Designed the paper structure and gave the feedbacks to all its versions. Discussed and enhanced the manuscript, revised the model and formulation, added sections to manuscript.

Ethics

We testify that this research paper is original and contains unpublished material.

References

Al-Dulaimy, A., A. Zekri, W. Itani and R. Zantout, 2016. Towards solving the problem of virtual machine placement in cloud computing: A job classification approach. J. Comput. Sci., 12: 113-127. DOI: 10.3844/jcssp.2016.113.127.

Banerjee, S., M. Adhikari, S. Kar and U. Biswas, 2015. Development and analysis of a new cloud*let al*location strategy for QoS improvement in cloud. Arabian J. Sci. Eng., 40: 1409-1425.

Bilal, K., O. Khalid, A. Erbad and U.K. Samee, 2018. Potentials, trends and prospects in edge technologies: Fog, cloudlet, mobile edge and micro data centers. Comput. Networks, 130: 94-120

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Generation Comput. Syst., 25: 599-616.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F.D. Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Practice Experience, 41: 23-50. DOI: 10.1002/spe.995

CGS, 2015. Cluster green software. http://www.clustergreensoftware.nl/english/

Chatterjee, T., V.K. Ojha, M. Adhikari, S. Banerjee and U. Biswas *et al*., 2014. Design and implementation of an improved datacenter broker policy to improve the QoS of a cloud. Proceedings of the 5th International Conference on Innovations in Bio-Inspired Computing and Applications, (ICA' 14), Springer International Publishing, pp: 281-290.

Cheng, H.K., Z. Li and A. Naranjo, 2016. Research note—Cloud computing spot pricing dynamics: Latency and limits to arbitrage. Inform. Syst. Res., 27: 145-165. DOI: 10.1287/isre.2015.0608

Chu, P.C. and J.E. Beasley, 1997. A genetic algorithm for the generalised assignment problem. Comput. Operations Res., 24: 17-23. DOI: 10.1016/S0305-0548(96)00032-9

Cook, G. and D. Pomerantz, 2015. Clicking clean: A guide to building the green internet.

Cook, G., 2012. How clean is your cloud. Catalysing an Energy Revolution.

Cook, G., 2014. Clicking clean: How companies are creating the green companies? Greenpeace Int. Tech. Rep.

Dong, Z., N. Liu and R. Rojas-Cessa, 2015. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. J. Cloud Comput., 4: 5-5.

Fan, Y., H. Ding, L. Wang and X. Yuan, 2016. Green latency-aware data placement in data centers. Comput Netw. 110: 46-57. DOI: 10.1016/j.comnet.2016.09.015

Fong, S., D. Suash and X.S. Yang, 2018. How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. Proceedings of the Progress in Intelligent Computing Techniques: Theory, Practice and Applications, (TPA' 18), Springer, Singapore, pp: 3-25.

Fox, A., R. Griffith, A. Joseph, R. Katz and A. Konwinski *et al*., 2009. Above the clouds: A Berkeley view of cloud computing. PhD thesis University of California, Berkeley.

Gartner, I., 2007. Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions.

Golberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edn., Addison-Wesley, Boston, ISBN-10: 0201015765, pp: 412.

Ha, M.T., M. Turilli, A. Merzky and S. Jha, 2018. Towards general distributed resource selection.

http://wintelguy.com/wanlat.html

https://docs.microsoft.com/enus/azure/guidance/guidance-compute-multipledatacenters

Huai, W., Z. Qian, X. Li, G. Luo and S. Lu, 2013. Energy aware task scheduling in data centers. JoWUA, 4: 18-38.

Kliazovich, D., S.T. Arzo, F. Granelli, P. Bouvry and S.U. Khan, 2013. e-STAB: Energy-efficient scheduling for cloud computing applications with traffic load balancing. Proceedings of the IEEE International Conference on and IEEE Cyber, Physical and Social Computing Green Computing and Communications (GreenCom), Aug. 20-23, IEEE Xplore Press, Beijing, China, pp: 7-13. DOI: 10.1109/GreenCom-iThings-CPSCom.2013.28

Koomey, J.G., 2007. Estimating total power consumption by servers in the US and the world.

Kumar, D., B. Sahoo, B. Mondal and T. Mandal, 2015. A genetic algorithmic approach for energy efficient task consolidation in cloud computing. Int. J. Comput. Applic.

Kumar, J. and A.K. Singh, 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Generation Comput. Syst., 81: 41-52. DOI: 10.1016/j.future.2017.10.047

Liu, S., S. Ren, G. Quan, M. Zhao and S. Ren, 2013. Profit aware load balancing for distributed cloud data centers. Proceedings of the IEEE 27th International Symposium on Parallel and Distributed Processing, May 20-24, IEEE Xplore Press, Boston, pp: 611-622. DOI: 10.1109/IPDPS.2013.60

Maiti, S. and P. Sudeep, 2017. DELCA: DVFS efficient low cost multicore architecture. Proceedings of the on Great Lakes Symposium on VLSI, May 10-12, Banff, Alberta, pp: 107-112. DOI: 10.1145/3060403.3060422

Mazumdar, P., S. Agarwal and A. Banerjee, 2016. Pro SQL Server on Microsoft Azure. 1st Edn., Apress, Berkeley, ISBN-10: 1484220838, pp: 211.

McKinsey, C., 2008. Revolutionizing data center efficiency. McKinsey Company.

Nemhauser, G.L. and L.A. Wolsey, 1988. Integer programming and combinatorial optimization. COAL Bull., 20: 8-12.

Piao, J.T. and J. Yan, 2010. A network-aware virtual machine placement and migration approach in cloud computing. Proceedings of the 9th International Conference on Grid and Cooperative Computing, Nov. 1-15, IEEE Xplore Press, Nanjing, pp: 87-92. DOI: 10.1109/GCC.2010.29

PLT, 2016. Planet lab traces. https://www.planet-lab.org

SPEC, 2016. Standard performance evaluation corporation. SPEC. http://www.spec.org

Sverdlik, Y., 2011. Growth in data center electricity use 2005 to 2010. Analytics Press, Tech. Rep.

Toosi, A.N., R.O. Sinnott and B. Rajkumar, 2018. Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka. Future Generation Comput. Syst., 79: 765-775. DOI: 10.1016/j.future.2017.05.042

Yin, Z.Y., Y.F. Jin, S.L. Shen and H.W. Huang, 2017. An efficient optimization method for identifying parameters of soft structured clay by an enhanced genetic algorithm and elastic–viscoplastic model. Acta Geotechnica, 12: 849-867. DOI: 10.1007/s11440-016-0486-0