Optimization of Virtual Resources Allocation in Cloud Computing Environment

Samson B. Akintoye
ISAT Laboratory, Department of Computer Science
University of the Western Cape
Cape Town, South Africa
sakintoye@uwc.ac.za

Antoine Bagula
ISAT Laboratory, Department of Computer Science
University of the Western Cape
Cape Town, South Africa
abagula@uwc.ac.za

***Abstract*—In cloud computing, the allocation of resources plays a key role in determining the performance, resource utilization and power consumption of the data center. The appropriate allocation of virtual machines in cloud data centers is also one of the important optimization problems in cloud computing, especially when the cloud infrastructure is made of lightweight computing devices. In this paper, we represent the resources' allocation problem in cloud computing environment as a linear programming model and propose a Hungarian Algorithm Based Binding Policy(HABBP) as a solution for optimizing the model. Finally, we propose an HABBP software implementation as contributed code to the popular CloudSim simulator and compared the HABBP performance to the conventional CloudSim binding policy and a binding based on the Simplex algorithm. Our simulation results show that the newly proposed policy outperforms the conventional binding policy implemented in the CloudSim in terms of jobs total execution time.**

***Index Terms*—Resources Allocation, Hungarian Algorithm, Virtual Machine, Cloud Computing, Simulation and CloudSim**

I. Introduction

Cloud computing is a new computing paradigm that provides on-demand network access to large pool of compute, storage and networking resources over the internet [15]. This paradigm offers cost savings because it is based on a pay-as-you-use model where customers pay for computing resources from a service provider only when needed and for only what they use. Cloud computing has three deployment models, namely: public cloud, private cloud, and hybrid cloud [15][16]. Furthermore, Cloud computing services are classified into three types as Infrastructure-as-a-service (IaaS), Platform- as-a-Service (PaaS) and Software-as-a-Service (SaaS). IaaS refers to providing hardware infrastructure such as CPU, memory and storage as a service. PaaS refers to providing platforms such as software development frameworks, operating systems or multi-tenant application supports as a service while in SaaS software and applications are provided as a service. These services are provided to the users through virtualization and distributed computing concepts. Virtualization is a technique by which physical infrastructures are abstracted to provide virtualized resources called virtual machines (VMs) for high-level applications [18]. VMs can be homogeneous or heterogeneous. For homogeneous VMs, VMs have the same number and size of CPUs, memory and storage while heterogeneous VMs have different number and sizes of CPUs, memory and storage [2]. In cloud computing, virtualization offers the capability of pooling computing resources from clusters of servers and assigning VMs to jobs on-demand. For each task received, either a new VM is initialized or it is assigned to an existing VM of the same user [16]. Once the task is executed, all the acquired resources are released to become parts of the free resource pool. However, due to the limited amount of physical resources available, resource allocation becomes a challenging issue for the cloud service providers (CSPs). A CSP decides on the number of VMs to be created based on the cloud user's request. In cloud computing environment, resource allocation range from mapping tasks to VMs and placement of VMs to physical Machines (PMs). The resource is allocated base on the Service Level Agreement (SLA) between the CSP and the cloud user. The SLA contains information about quality of service (QoS) to be provided to the user in terms of various performance parameters such as throughput, reliability, blocking probability and response time, the payment process and SLA violation penalty [4]. The goal of the CSP is to maximize profit and resource utilization while the goal of cloud user is to minimize the cost of leasing the resources. In this paper, we assume that the tasks and virtual resources are heterogeneous and physical resources are homogeneous. We also assume that the number of available physical resources are limited compared to the number of cloud user on-demand requests. We propose a model for mapping the tasks (cloudlets) to VMs with the aim of maximizing the processing speed or minimizing the execution time of the tasks. The major contributions of this paper are as follows:

- **Mathematical modelling**. We represent the virtual resource allocation problem in cloud computing environment as a linear programming model minimizing the resources allocation cost in a setting where multiple cloud user requests have to be processed on a limited number of physical resources.
- **A binding policy**. We propose the Hungarian algorithm as a heuristic solution to the the linear programming model and use the algorithm to implement a novel binding policy for the popular CloudSim simulator.
- **CloudSim implementation**. We propose the HABBP module as a contributed module to CloudSim which

978-1-5386-2775-4/17/$31.00 ©2017 IEEE

comprises i) a graphical user interface as a front-end component enabling cloud users to interact with the CloudSim and to configure the parameters for tasks, VM and PM from the interface rather than embedding parameters in the CloudSim source codes and ii) a novel binding method as a back-end component.

- **Performance evaluation.** We compare the proposed binding policy to the conventional binding policy implemented by the CloudSim simulator and benchmark both solution against the Simplex algorithm commonly used as linear solver.

The remainder of this paper is organized as follows. Virtualization concepts are discussed in section 2 while section 3 presents related work as regards to resource allocation problem in cloud computing. The linear programming model for cloud resources allocation problem is proposed in section 4 and section 5 presents a Hungarian Algorithm Based Binding Policy (HABBP) as solution for the optimization of the model. Section 6 presents the implementation of the policy in terms of the developed graphical user interface and the CloudSim method used to implement the policy. Furthermore, the simulation results are included in this section in terms of binding policy comparison between the proposed HABBP and tim conventional binding policy. Our conclusions and avenues for future work are presented in section 7.

II. Related Work

Many researchers have proposed different models and algorithms to optimize the utillization of resources in cloud computing environment. In [13], the authors proposed a stochastic model of a cloud computing cluster in which jobs arrive according to a stochastic process and request heterogeneous virtual machines (VMs). A non-preemptive algorithm is used for load balancing among servers, and for scheduling VM configurations. To reduce the communication overhead, a more distributed system is also considered where each server maintains its own queues. The simulation results show that there is slight difference in delay performance between centralized and distributed queuing systems. They also show that the non-preemptive algorithm used is better than the widely-used Best-Fit scheduling algorithm in term of throughput. Hallawi et al.[7] proposed two genetic algorithms to find optimal solutions for multi-dimensional vector bin packing problems with the goal to improve cloud resource allocation and Virtual Machines (VMs) consolidation. Two algorithms, namely Combinatorial Ordering First-Fit Genetic Algorithm (COFFGA) and Combinatorial Ordering Next Fit Genetic Algorithm (CONFGA) have been developed for that and combined. The proposed hybrid algorithm targets to minimise the total number of running servers and resources wastage per server. The solutions obtained by the new algorithms are compared with latest solutions from literature. The experimental results show that the proposed algorithm COFFGA outperforms other previous multi-dimension vector bin packing heuristics such as Permutation Pack (PP), First Fit (FF) and First Fit Decreasing (FFD) by 4%, 34%, and 39%, respectively. It also achieved better performance than the existing genetic algorithm for multi-capacity resources virtual machine consolidation (RGGA) in terms of performance and robustness. Lin [10] discusses various resource scheduling algorithms, and groups the algorithms based on some factors, such as time, cost, and energy. The benefit of the paper is to assist cloud users in selecting suitable scheduling algorithms based on their service needs. In [11], the authors proposed a directed acyclic graph based scheduling algorithm called earliest finish time duplication algorithm to schedule multiple tasks in heterogeneous cloud computing environments. They showed that by preprocessing the cloud resources before scheduling, the proposed algorithm has better performance in terms of task scheduling time than the popular heterogeneous earliest finish time algorithms. S. T. Maguluri et. al. [12] proposed heavy traffic optimal resource allocation algorithms for cloud computing clusters. The authors studied a stochastic model of cloud computing, where jobs arrive according to a stochastic process and request resources like CPU, memory and storage space. A model where the resource allocation problem can be separated into a routing or load balancing problem and a scheduling problem is considered. Join-the-shortest-queue routing and power of-two-choices routing algorithms with MaxWeight scheduling algorithm is studied. In the algorithms Jobs are first routed to the servers, and are then queued at the servers, and a scheduler schedules jobs at each server. It is established by using diffusion limit arguments that the power-of- two-choices algorithm is heavy traffic optimal. Lakra et al. [9] proposed a multi-objective task scheduling algorithm for mapping tasks to a VMs in order to improve the throughput of the datacenter and reduce the cost without violating the SLA (Service Level Agreement) for an application in cloud SaaS environment.

III. Resource allocation Problem Model

We formulate a linear programming problem model for binding tasks (also known as cloudlets) to virtual resources known as *VMs*. We consider a set of *VM* denoted by $vm_i : i \leq n$ with $n > 1$. We also consider a set tasks (cloudlets) associated with each on-demand user request (job) denoted by $t_j : j \leq m$ with $m > 1$. We assume that each virtual machine execute only one cloudlet and each cloudlet need be allocated to only one virtual machine.

Let $n = m$ and $C = [c_{ij}]$ be any $n \times n$ matrix in which c_{ij} is the cost of allocating virtual machine i to cloudlet j;

$$c_{ij} = \frac{t_j}{vm_i} \tag{1}$$

Let $X = [x_{ij}]$ be the $n \times n$ matrix where

$$x_{ij} = \begin{cases} 1 & \text{, if virtual machine } i \text{ is allocated to cloudlet } j \\ 0 & \text{, if virtual machine } i \text{ is not allocated to cloudlet } j \end{cases} \tag{2}$$

Our objective is to optimize the total cost $p(X)$, defined as sum of the cost of allocating cloudlets to the available virtual machines. We formulate an optimization problem as an linear programming model in terms of a function p as:

$$minimize\ p(X) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \tag{3}$$

subject to the constraints

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for j = 1, 2,...n} \ . \tag{4}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \text{for i = 1, 2,...m} \ . \tag{5}$$

$$x_{ij} = \quad 0 \text{ or } 1 \ . \tag{6}$$

Hence any matrix satisfying (4) and (5) is a solution and corresponds to a permutation σ of of a set $N = \{1, 2, ..., n\}$ derived by setting $\sigma(i) = j$ if and only if $x_{ij} = 1$. Also, if X is a solution corresponding to σ, then

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} = c_{i\sigma(i)} \tag{7}$$

Summing over i from 1 to n, we obtain

$$\sum_{i=1}^{n} c_{i\sigma(i)} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{8}$$

Thus, any solution X on which $p(X)$ is minimum is known as optimal solution. We can transform a given allocation problem specified by C into another one specified by a matrix $\overline{C} = [\overline{c}_{ij}]$, such that $\overline{c}_{ij} \geq 0$, for all pairs i, j, where the two problems have the same set of optimal solutions. If X^* is an optimal solution to the problem specified by $\overline{C}$, then must also be an optimal solution to the one specified by C. Theorem 1 explains how we can transform a matrix into another one which has the same set of optimal solutions.

Theorem 1 A solution X is an optimal solution for

$$p(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

if and only if it is an optimal solution for

$$\overline{p}(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{c}_{ij}x_{ij}$$

where $\overline{c}_{ij} = c_{ij} - u_i - v_j$ for any of $u_1, ..., u_n$ and $v_1, ..., v_n$ and u_i and v_j are real numbers for all i and j.

Proof: We would establish that the difference between the functions $p(X)$ and $\overline{p}(X)$ is constant $\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$.

$$\overline{p}(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{c}_{ij}x_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} (c_{ij} - u_i - v_j)x_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} v_j x_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} - \sum_{j=1}^{n}\sum_{i=1}^{n} v_j x_{ij}$$

$$= p(X) - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} x_{ij}$$

From (4) and (5),

$$= p(X) - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j$$

This shows that, $p(X) - \overline{p}(X) = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$

Therefore, a solution X minimizes $p(X)$ if and only if it minimizes $\overline{p}(X)$.

IV. Hungarian Algorithm Based Binding policy(HABBP)

In this section, we present Hungarian Algorithm for solving tasks allocation problem in cloud computing environment.

A. Notation

- Given the CostMatric *(cm)* of the size n x m
- n is number of VMs
- m is the number of cloudlets.
- cm_{ij} denote the time required to finish cloudlet i by virtual machine j.

B. The algorithm

Algorithm 1 shows the pseudo-code of the Hungarian Algorithm for task allocation in cloud computing environment. First, we initialize different variables (Steps 1-4). We initialize *CostMatrix* by dividing the length of cloudlet with the MIPS of virtual machine. If the number of cloudlets and number of virtual machines are not the same then we add the dummy cloudlets/virtual machines to make *CostMatrix* square matrix. Then we compute the *reducedCostMatrix* by subtracting minimum value of each row and column from the row and column *CostMatrix*. After that, we compute *lineCostMatrix*. If the number of lines is not equal to the number of virtual machines, then we subtract the minimum uncovered element to every covered element. If an element is covered twice, add the minimum element to it.
Finally, We apply the mapping to the original matrix, disregarding dummy rows. and we add the cost of binding cloudlet to virtual machine to give total minimum cost C.

C. Example

We illustrate through an example the concept of the HABBP and how it can be used to optimize the allocation of virtual resources in cloud computing environment. Table I shows three cloudlets in the queue with broker and table II shows virtual machines created in the cloud computing environment.

Algorithm 1 calculate total allocation cost C

input:
n: number of virtual machines
m: number of cloudlets
t_i : list of cloudlet $(i \in 1..m)$
vm_j : list of virtual machine $(j \in 1..n)$
output:
C: total allocation cost

1: /* $Initialize\ and\ populace\ CostMatrix$ */
2: **for all** i in set $(1..n)$ **do**
3: **for all** j in set $(1..m)$ **do**
4: cm_{ij} = t_i / vm_j
5: **end for**
6: **end for**
7: **if** $n \neq m$ **then**
8: add dummy cloudlets with 0 execution time value to make CostMatrix square matrix
9: **end if**
10: mr = minimum row element
11: mc = minimum column element
12: /* $compute\ the\ reduced\ CostMatrix$ */
13: **for all** j in set $(1..n)$ **do**
14: cm_{nj} = cm_{nj} - mr
15: **end for**
16: **for all** i in set $(1..n)$ **do**
17: cm_{ni} = cm_{ni} - mc
18: **end for**
19: /* $compute\ the\ lineCostMatrix$ */
20: l_n = minimum-number-line()
21: **if** $l_n < m$ **then**
22: **for all** j in set $(1..n)$ **do**
23: **for all** j in set $(1..n)$ **do**
24: **if** element are uncovered **then**
25: cm_{ij} = cm_{ij} - $min(uncoveredElement)$
26: **else if** element are covered by two line **then**
27: cm_{ij} = cm_{ij} + $min(uncoveredElement)$
28: **end if**
29: **end for**
30: **end for**
31: **end if**
32: /* $find\ the\ mapping$ */
33: apply the matching to the original matrix, disregarding dummy rows.
34: adding the costs will give the total minimum cost
35: **return** total allocation cost C

TABLE I: Cloudlet specifications

	cloudlet 1	cloudlet 2	cloudlet 3
id	0	1	2
file size	500	1000	1000
length	20000	60000	90000
output size	500	2048	2048

The algorithm works as follows: We initialize *CostMatrix* by dividing length of cloudlet with the MIPS of virtual machine as shown in the Table III.

TABLE II: virtual machine specifications

	virtual machine 1	virtual machine 2	virtual machine 3
vmid	0	1	2
size	1000	1000	1000
MIPS	200	500	250
ram	2048	2048	2048
pesNumber	1	2	2
bandwidth	500	500	500

TABLE III: Initialize CostMatrix

	cloudlet 1	cloudlet 2	cloudlet 3
vm 1	100	300	450
vm 2	40	120	180
vm 3	80	240	360

In this case, the number of cloudlets is equal to the number of virtual machines. Therefore, we don't need to add the dummy cloudlet/virtual machine values to make *CostMatrix* square matrix. We compute the *reducedCostMatrix* by subtracting minimum value of each row and column from the row and column *CostMatrix* give table IV and V respectively.

TABLE IV: Row reducedCostMatrix

	cloudlet 1	cloudlet 2	cloudlet 3
vm 1	0	200	350
vm 2	0	80	140
vm 3	0	160	280

TABLE V: Column reducedCostMatrix

	cloudlet 1	cloudlet 2	cloudlet 3
vm 1	0	120	210
vm 2	0	0	0
vm 3	0	80	140

Furthermore, we compute *lineCostMatrix*, this denotes lines that cover all zeros in *reducedCostMatrix*. In this example, there are two lines. The lines are on column 1 and row 2 of *reducedCostMatrix*. Since the number of line is not equal to the number of virtual machines, we subtract minimum of all uncovered elements from all uncovered elements as indicated in Table VI

TABLE VI: reducedCostMatrix

	cloudlet 1	cloudlet 2	cloudlet 3
vm 1	0	40	130
vm 2	0	0	0
vm 3	0	0	60

Again, we compute the minimum number of lines required to cover all zeros in the matrix. The lines are on cloumn 1 , row 2 and row 3 of *reducedCostMatrix*. Since number of

lines is equals to the number of virtual machines, an optimal allocation exists among the zeros in the *reducedCostMatrix*. Therefore, cloudlet 1 allocated to virtual machine 1, cloudlet 3 allocated to virtual machine 2, and cloudlet 2 allocated to virtual machine 3.

TABLE VII: Optimal Allocation

cloudlets	virtual machines
cloudlets 1	vm 1
cloudlets 2	vm 3
cloudlets 3	vm 2

The total cost of the optimal allocation of cloudlets to virtual machine is to100s + 180s + 240s = 520s.

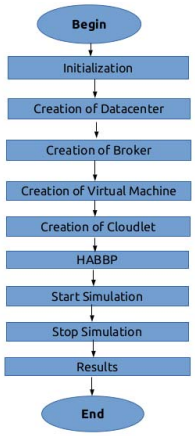


Fig. 1: CloudSim Life cycle with HABBP

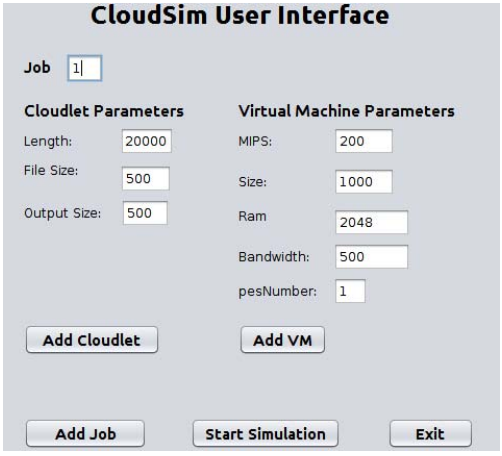


Fig. 2: CloudSim User Interface

V. Experiments and Results

This section presents the experimental results obtained by testing the model on an Intel(R) Core(TM) i5-4590 machine with 3.30Ghz CPU and 8GB RAM. Our experiments are based on the open-source cloud stimulator called CloudSim [5] and netbean IDE 8.2. CloudSim is an extensible Java-based open source simulation toolkit that provide support for modelling and simulation of cloud computing environments. CloudSim is an advanced simulator cloud computing environments with great properties such as scaling well and has a low simulation overhead [6]. It provides classes for data centers, virtual machines, applications, users, computational resources, and scheduling policies. As shown in Fig. 1, there are different stages of CloudSim life cycle range from initialization of cloud infrastructures to the simulation results. Our goal was to validate the performance gains derived from the HABBP policy compared to the conventional task binding policy implemented in the CloudSim.

TABLE VIII: List of Jobs

Cloudlet ID	Job 1 (Length)	Job 2 (Length)	Job 3 (Length)	Job 4 (Length)	Job 5 (Length)
0	20000	30000	150000	40000	200000
1	60000	50000	80000	20000	80000
2	90000	110000	130000	80000	160000
3	40000	10000	90000	30000	20000
4	120000	70000	10000	10000	40000
5	200000	20000	40000	90000	90000
6	70000	80000	120000	110000	120000
7	80000	40000	60000	60000	10000
8	50000	160000	20000	150000	130000
9	10000	90000	70000	200000	100000
10	150000	350000	45000	75000	5000
11	250000	250000	250000	300000	25000
12	45000	100000	85000	450000	155000
13	25000	95000	140000	87000	95000
14	75000	25000	100000	250000	4000
15	30000	85000	35000	50000	110000
16	300000	180000	130000	100000	210000
17	55000	35000	75000	130000	55000
18	65000	15000	95000	95000	85000
19	85000	9000	200000	400000	350000

TABLE IX: List of Virtual Machines

VM ID	MIPS Parameter
0	1000
1	500
2	200
3	2000
4	250
5	100
6	50
7	125
8	150
9	400
10	1500
11	350
12	450
13	600
14	700
15	850
16	900
17	550
18	1200
19	300

TABLE X: Results of Conventional binding policy in CloudSim

Cloudlet ID	Job 1		Job 2		Job 3		Job 4		Job 5	
	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time
0	0	20	0	30	0	150	0	40	0	200
1	1	120	1	100	1	160	1	40	1	160
2	2	450	2	550	2	650	2	400	2	800
3	3	20	3	5	3	45	3	15	3	10
4	4	480	4	280	4	40	4	40	4	160
5	5	2000	5	200	5	400	5	900	5	900
6	6	1400	6	1600	6	2400	6	2200	6	2400
7	7	640	7	320	7	480	7	480	7	80
8	8	333	8	1066	8	133	8	1000	8	866
9	9	25	9	225	9	175	9	500	9	250
10	10	100	10	233	10	30	10	50	10	3
11	11	714	11	714	11	714	11	857	11	71
12	12	100	12	222	12	189	12	1000	12	344
13	13	42	13	158	13	233	13	145	13	158
14	14	107	14	36	14	143	14	357	14	6
15	15	35	15	100	15	41	15	59	15	129
16	16	333	16	200	16	144	16	111	16	233
17	17	100	17	64	17	136	17	236	17	100
18	18	54	18	13	18	79	18	79	18	71
19	19	283	19	30	19	667	19	1333	19	1167
Total Exec. Time	7356		6147		7009		9842		8109	

TABLE XI: Results of HABBP in CloudSim

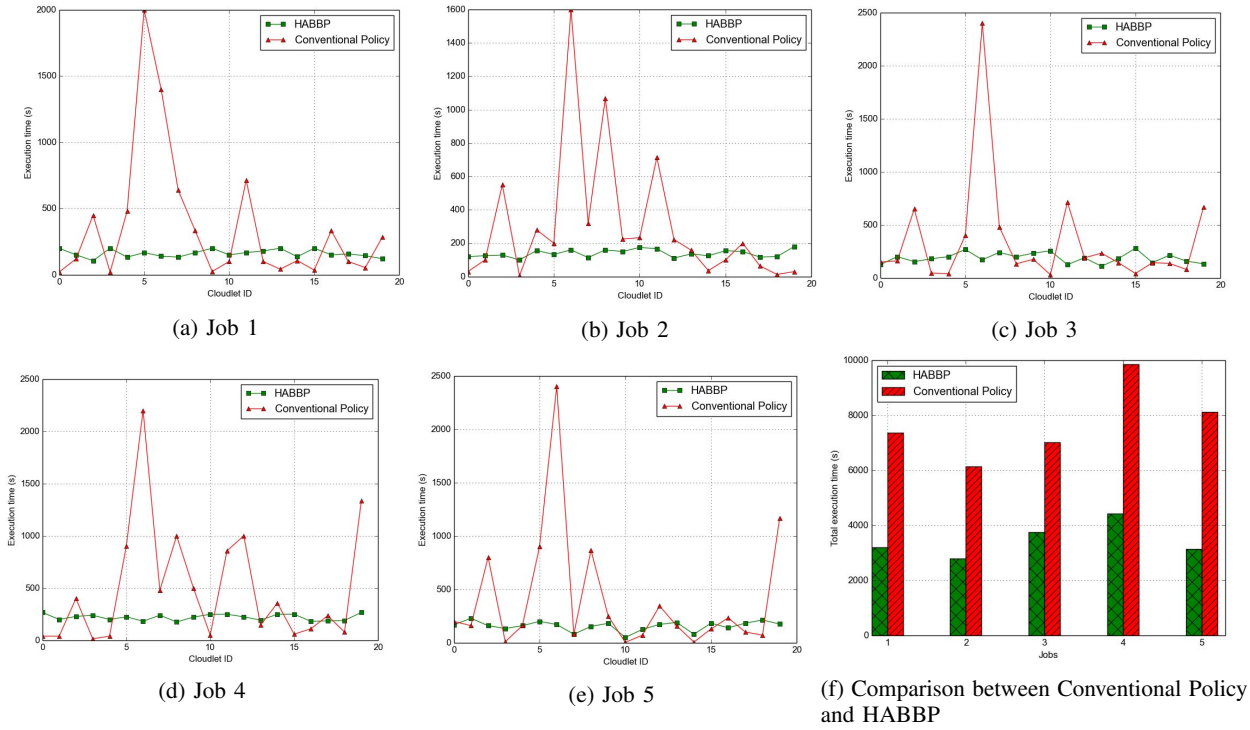
Cloudlet ID	Job 1		Job 2		Job 2		Job 4		Job 5	
	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time	VM ID	Exec. Time
0	5	200	4	120	18	125	8	267	18	167
1	9	150	9	125	9	200	5	200	11	229
2	15	106	15	129	15	153	11	229	0	160
3	2	200	5	100	1	180	7	240	8	133
4	16	133	12	156	6	200	6	200	4	160
5	18	167	8	133	8	267	9	225	12	200
6	1	140	1	160	14	171	13	183	14	171
7	13	133	11	114	4	240	4	240	7	80
8	19	167	0	160	5	200	15	176	15	153
9	6	200	13	150	19	233	16	222	17	182
10	0	150	3	175	2	255	19	250	5	50
11	10	167	10	167	3	125	18	250	2	125
12	4	180	16	111	12	189	3	225	16	172
13	7	200	14	136	0	110	12	193	1	190
14	17	136	2	125	17	182	0	250	6	80
15	8	200	17	155	7	280	2	250	13	183
16	3	150	18	150	16	144	17	182	10	140
17	11	157	19	117	11	214	14	186	19	183
18	12	144	7	120	13	158	11	190	9	213
19	14	121	6	180	10	133	10	267	3	175
Total Execution Time	3201		2783		3759		4425		3146	

TABLE XII: Policies computational time

HABBP	Conventional	Simplex algorithm
0.0157	0.0019	0.0198
0.0216	0.0182	0.0318
0.0178	0.0097	0.0290
0.0166	0.0042	0.0200
0.0174	0.0073	0.0256

A. *Implementation of the Proposed HABBP*

The major drawback of the current CloudSim is the lack of a Graphical User Interface(GUI) that would allow cloud users to configure cloudlets and the cloud infrastructure parameters and the lack of optimal cloudlets-to-virtual machines binding policies. In the present work, we extended CloudSim to i) implement and integrate a graphical user interface using a java *Jframe* class as shown in see fig. 2 and ii) introduce a new cloudlets-to-virtual machines binding policy in the cloud computing environment by creating a new method called *HungarianAlgorithmBinding()* in the *DatacenterBroker* class of CloudSim.



(a) Job 1 (b) Job 2 (c) Job 3

(d) Job 4 (e) Job 5 (f) Comparison between Conventional Policy and HABBP

Fig. 3: Execution time of the Cloudlets of Jobs 1, 2, 3, 4 and 5

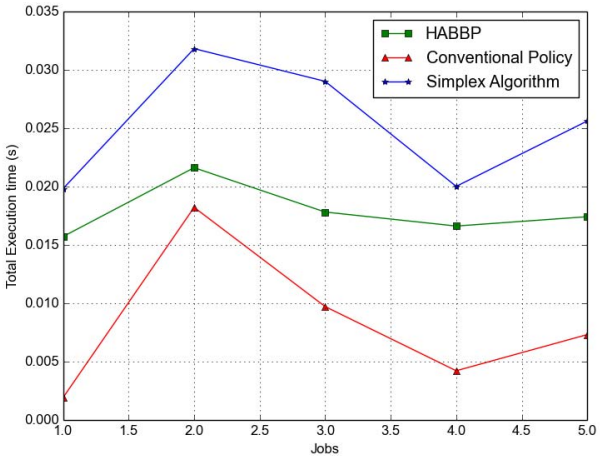
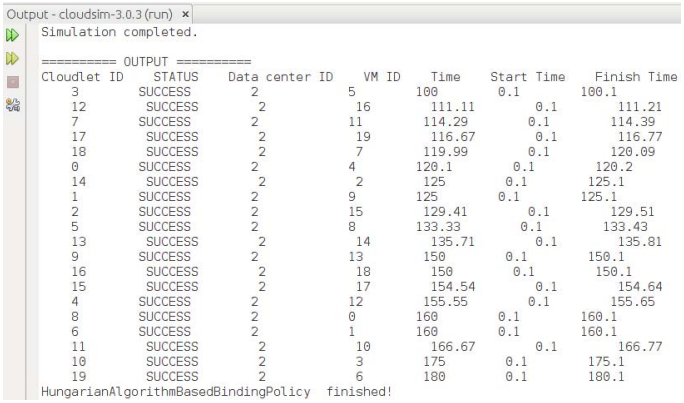


Fig. 4: Policies computational time



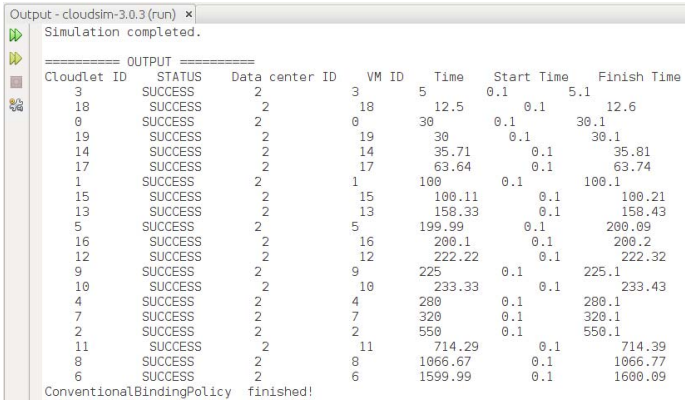
```
Output - cloudsim-3.0.3 (run)
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
     3         SUCCESS         2            5       100        0.1          100.1
     12        SUCCESS         2            16      111.11     0.1          111.21
     7         SUCCESS         2            11      114.29     0.1          114.39
     17        SUCCESS         2            19      116.67     0.1          116.77
     18        SUCCESS         2            7       119.99     0.1          120.09
     0         SUCCESS         2            4       120.1      0.1          120.2
     14        SUCCESS         2            2       125        0.1          125.1
     1         SUCCESS         2            9       125        0.1          125.1
     2         SUCCESS         2            15      129.41     0.1          129.51
     5         SUCCESS         2            8       133.33     0.1          133.43
     13        SUCCESS         2            14      135.71     0.1          135.81
     9         SUCCESS         2            13      150        0.1          150.1
     16        SUCCESS         2            18      150        0.1          150.1
     15        SUCCESS         2            17      154.54     0.1          154.64
     4         SUCCESS         2            12      155.55     0.1          155.65
     8         SUCCESS         2            0       160        0.1          160.1
     6         SUCCESS         2            1       160        0.1          160.1
     11        SUCCESS         2            10      166.67     0.1          166.77
     10        SUCCESS         2            3       175        0.1          175.1
     19        SUCCESS         2            6       180        0.1          180.1
HungarianAlgorithmBasedBindingPolicy  finished!
```

Fig. 5: Mapping Cloudlets to VMs using HABBP

B. Experimental results

Our experiments were conducted by allocating each virtual machine to different host machines of the same capacity and all host machines are located in the same datacenter. Thereafter, we simulated five jobs using HABBP, conventional binding policy and Simplex algorithm to allocate cloudlets to VMs in the CloudSim. Each job has 20 cloudlets and assigned to heterogeneous VMs. Each cloudlet and VM had different lengths and MIPS values. Other specifications such as file size, and output size values of all cloudlets and size, ram, bandwidth, and pesNumber of all VMs were constant as shown in the table VIII and IX.

The simulation results are presented in Tables X and XI, Figs. 5 and 6, and plotted in Fig. 3 and 4. In conventional binding policy, cloudlets are mapped to the VMs sequentially, that is cloudlet ID 0 to VM ID 0, cloudlet ID 1 to VM ID 1, cloudlet ID 2 to VM ID 2, cloudlet ID 3 to VM ID 3 , cloudlet ID 4 to VM ID 4 and etc. On the other hand, HABBP allocates cloudlets to VMs based on the operations in the HABBP. For example, cloudlet ID 0 allocated VM ID 4, cloudlet ID 1 to VM ID 9, cloudlet ID 2 to VM ID 15, cloudlet ID 3 to VM ID 5, cloudlet ID 4 to VM ID 12 and etc. in the job 2.



```
Output - cloudsim-3.0.3 (run) ×
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
     3         SUCCESS        2            3        5        0.1          5.1
     18         SUCCESS        2            18       12.5       0.1          12.6
     0         SUCCESS        2            0        30        0.1          30.1
     19         SUCCESS        2            19       30        0.1          30.1
     14         SUCCESS        2            14       35.71        0.1          35.81
     17         SUCCESS        2            17       63.64        0.1          63.74
     1         SUCCESS        2            1        100        0.1          100.1
     15         SUCCESS        2            15       100.11        0.1          100.21
     13         SUCCESS        2            13       158.33        0.1          158.43
     5         SUCCESS        2            5        199.99        0.1          200.09
     16         SUCCESS        2            16       200.1        0.1          200.2
     12         SUCCESS        2            12       222.22        0.1          222.32
     9         SUCCESS        2            9        225        0.1          225.1
     10         SUCCESS        2            10       233.33        0.1          233.43
     4         SUCCESS        2            4        280        0.1          280.1
     7         SUCCESS        2            7        320        0.1          320.1
     2         SUCCESS        2            2        550        0.1          550.1
     11         SUCCESS        2            11       714.29        0.1          714.39
     8         SUCCESS        2            8        1066.67        0.1          1066.77
     6         SUCCESS        2            6        1599.99        0.1          1600.09
ConventionalBindingPolicy  finished!
```

Fig. 6: Mapping Cloudlets to VMs using conventional binding policy

We also compared the overall performance of the HABBP with the conventional binding policy and benchmarked both solutions against the Simplex algorithm in terms of the execution time of cloudlets in each job and total execution time of individual jobs. In Figs. 3(a), 3(b), 3(c), 3(d) and 3(e), some cloudlets take slightly longer time to complete in HABBP than conventional policy while some other cloudlets take very significant longer time to complete in conventional policy than HABBP. However, Fig. 3(f) where the total execution time performance of different jobs for HABBP and conventional policy is presented, shows that HABBP constantly outperforms the conventional policy. Take job 2 as an example. Compared to the default policy, the total execution time for HABBP is reduced by 54.73%. HABBP and Simplex algorithm give the same jobs execution time. However, HABBP has better performance that Simplex algorithm in term of computational time as shown in table XII and Fig. 4.

VI. Conclusion and Future Work

In this paper, we have revisited the issue of resource allocation in cloud computing environments and proposed HABBP binding policy as a new and optimal solution to the virtual resources allocation problem. HABBP uses a load balancing policy for binding cloudlets to virtual machines in such a way that each cloudlet is allocated to the appropriate virtual machine to optimize the total execution time of completing on-demand user requests. We have formulated an optimization model, proposed HABBP, simulated both HABBP and conventional binding policy in CloudSim and benchmarked these solutions against the Simplex algorithm.

The simulation results produced by our contributed code to the CloudSim simulation revealed the relative efficiency of the newly proposed HABBP policy in solving and optimizing the virtual resources allocation problem in the cloud computing environment. In a near future, HABBP will be used to optimize resource allocation in fog-based platforms targeting smart parking infrastructures [1], [8] in urban areas and drought mitigation [14] in the rural areas of Africa. For such deployments, the policy will be extended to account for the outdoor characteristics of the cloud computing network [17] and white space frequency bands [3] as they can heavily impact the access to the cloud nodes and thus influence the QoS provided by the cloud. The newly proposed policy will be implemented in these infrastructures using the widely know OpenStack cloud management platform.

References

[1] A. Bagula, L. Castelli, and M. Zennaro. On the design of smart parking networks in the smart cities: An optimal sensor placement model. *Sensors*, 15(7):15443 – 15467, 2015.

[2] J. S. Brian, Chee, and C. F. Jr. Cloud computing: technologies and strategies of the ubiquitous data center, crc, new york. 2010.

[3] T. X. Brown, E. Pietrosemoli, M. Zennaro, A. Bagula, H. Mauwa, and S. M. Nleya. A survey of tv white space measurements. *In International Conference on e-Infrastructure and e-Services for Developing Countries, Springer*, 15(7):164 – 172, 2014.

[4] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer System*, pages 559 – 616, 2009.

[5] R. N. Calheiros, R. Ranjan, and A. Belo. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper*, 41:23 – 50, 2011.

[6] S. Garg and R. Buyya. Networkcloudsim: modelling parallel applications in cloud simulations. *in: Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, pages 105 – 113, 2011.

[7] H. H. Huda Hallawi, Jorn Mehnen. Multi-capacity combinatorial ordering ga in application to cloud resources allocation and efficient virtual machines consolidation. *Future Generation Computer Systems.*, 69:1 – 10, 2017.

[8] E. M. Karbab, D. Djenouri, and A. Bagula. Car park management with networked wireless sensors and active rfid. *In IEEE International Conference on Electro/Information Technology. IEEE*, 2015.

[9] A. V. Lakra and D. K. Yadav. Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. *International Conference on Intelligent Computing, Communication & Convergence (ICCC)*, 48:107 – 113, 2015.

[10] C. T. Lin. Comparative based analysis of scheduling algorithms for rm in cloud computing environment. *International journal of Computer Science Eng*, 1(1):17 – 23, 2013.

[11] Z. Liu, W. Qu, W. Liu, Z. Li, and Y. Xu. Resource preprocessing and optimal task scheduling in cloud computing environments. *Concurrency Computat.: Pract. Exper. Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3204*, 2014.

[12] S. T. Maguluri, R. Srikant, and L. Ying. Heavy traffic optimal resource allocation algorithms for cloud computing clusters. In *24th International Teletraffic Congress*, number 25, 2012.

[13] S. T. Maguluri, R. Srikant, and L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters. In *in the Proceeding of IEEE INFOCOM*, pages 702 – 710, 2012.

[14] M. Masinde and A. Bagula. A framework for predicting droughts in developing countries using sensor networks and mobile phones. *In Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. ACM*, pages 390 – 399, 2010.

[15] P. Mell and T.Grance. The nist definition of cloud computing (draft). *NIST special publication: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf*, 800:145, September 2011.

[16] B. P. Rimal, E. Choi, and I. Lumb. A taxonomy and survey of cloud computing systems. *In Fifth international joint conference on INC, IMS and IDC*, pages 44 – 51, June 2009.

[17] M. Zennaro, H. Ntareme, and A. Bagula. Experimental evaluation of temporal and energy characteristics of an outdoor sensor network. *In Proceedings of the International Conference on Mobile Technology, Applications, and Systems, Article 99. ACM*, 2008.

[18] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7 – 18, January 2010.