Efficient and scalable ACO-based task scheduling for green cloud computing environment

Ado Adamou Abba Ari*‡, Irépran Damakoa†, Chafiq Titouna*§, Nabila Labraoui¶ and Abdelhak Gueroui*
*LI-PaRAD Laboratory, Université Paris Saclay, University of Versailles Saint-Quentin-en-Yvelines, France
Email: ado-adamou.abba-ari@uvsq.fr
†Department of Mathematics and Applied Computer Science, University of Ngaoundéré, Cameroon
‡Department of Mathematics and Computer Science, University of Maroua, Cameroon
§Department of Computer Science, University of Batna 2, Algeria
¶STIC Laboratory, University of Tlemcen, Algeria

***Abstract*—Cloud Computing has emerged as a popular technology that support computing on demand services by allowing users to follow the pay-per-use-on-demand model. Minimizing energy consumption in cloud systems has many benefits that enable green computing. Energy aware task scheduling in cloud to the users by service cloud providers has non negligible influences on optimal resources utilization and thereby on the cost benefit. The traditional algorithms for task scheduling are not well enough for cloud computing. In such environment, tasks should be efficiently scheduled such a way that the makespan is reduced. In this paper, we proposed a biologically inspired scheduling scheme, which is a based on a modified version of the ant colony optimization that aims at reducing the makespan time while ensuring load balancing among resources in order to enable green computing. Experiments of the proposed scheme in various scenario have been conducted in order to elaborate the impact of proposed models in the reduction of makespan. The obtained results demonstrate the effectiveness of the proposal in regards to the compared algorithms.**

***Keywords*—Green cloud computing, Task scheduling, modeling, ACO, CACO, Virtualization, Makespan, Cloudlets**

I. Introduction

A. Background

Cloud Computing, the Internet based computing that won a lot of momentum for its flexibility and elasticity, provides shared data and processing resources as services allowing users to not need to have their own infrastructure and follows the pay-as-you-go model. The cloud system has the potential to transform a large part of the IT industry since it makes software even more attractive as a service and shaping the way in which IT hardware is designed and purchased [1]. This system aims at power the next generation data centers with both applications delivered services through the Internet and the hardware as well as software in order to face with the growing requirement of user Quality of Service (QoS) and Quality of Experience (QoE). These services are generally computer applications, software development platforms as well as computing and storage resources. These services have long been referred as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Nowadays, billion of people continuously use cloud services that include online gaming, social networking, web hosting, content delivery, streamed contents as well as scientific applications. Each of these applications presents different features that include configuration and deployment requirements. Moreover, cloud providers are continuously looking techniques for allowing continuous services providing and of course for maximizing their profit while allowing a good QoS/QoE. Furthermore, the cloud system fundamentally relies on virtualization, especially on virtual machines. Therefore, due to the heterogeneity of computer and storage in the cloud infrastructure scheduling of tasks upon virtual machines is a crucial task [2], [3]. Due to the user's high demand, the cloud data centers are extremely challenging. The scheduling in cloud based requires quantifying the need of performance for resources allocation for different application and service models under different conditions. Moreover, it is necessary to avoid the presence of powered on machines when they are not being utilized. Hence, a good scheduling of task in cloud based environments leads to energy conservation and therefore, it may allow a green computing paradigm [4].

Thereby, to meet requirements of both users and cloud providers, techniques that allow the reduction of the overall time taken to complete the users' tasks as well as resources utilization, are needed. Scheduling refers to the set of policies that aim at controlling the order of computing task to be performed by a computer or a set of computers. There has been various works achieved in the literature on scheduling algorithm in various computing systems. However, limited research have been done so far in scheduling in cloud environment. Scheduling algorithms inspired by the swarm intelligence techniques such as social behavior of ant and honeybees colonies have proven their reputations in better solving complex optimization problems [5], [6].

We focused in this paper on the user-level task scheduling in cloud based environment. The proposal called CACO for Cloud-ACO mainly use the Ant Colony Optimization (ACO) features for making the optimal decision in the task scheduling problem. The main aim is minimizing the overall completion time while enabling better resource utilization. Therefore, in order to achieve a better load balancing across all the resources in the cloud system, we proposed a scheduler optimizer by exploiting the Max-Min Ant System (MMAS) features to propose some models that optimize the task scheduling compared to existing methods.

978-1-5386-3684-8/17 $31.00 © 2017 IEEE
DOI 10.1109/SmartCloud.2017.17

B. Authors' contributions

This paper contains an original contribution in the field of task scheduling in cloud based environment and highlights the adopted optimization scheme for maintaining a balanced load. We formulated the problem of task scheduling in user-level cloud computing system like an optimization problem. We proposed some models inspired by the ant system, which govern the optimal use of available resources in cloud data centers. In short, our main contributions can be summarized as follows:

- Formulation of the task scheduling problem in the cloud based environment;
- Proposition of the scheme based on the ant system for scheduling jobs;
- Simulation of the proposal in order to highlight it effectiveness;
- Comparison of the CACO with other solutions in order to show that, our scheduler delivers a reduced makespan.

C. Organization of the paper

The rest of the paper is organized as follows: section II presents a brief review of some related works; section III presents the task scheduling problem formulation; our scheduler optimizer based on the ant system is presented in section IV by first describing the scheme design and then the proposed CACO algorithm is described; this is followed by the performance evaluation and discussion in section V; and the conclusion and directions for future work are presented in section VI.

II. Related work

During the last few years, task scheduling problems related to Grid Computing environments have been investigated extensively by many researchers. However, task scheduling in green cloud computing environments for improving the load balancing has not been widely studied. Moreover, a number biologically inspired solution such as genetic algorithms (GA) and ACO have been applied to solve the task scheduling problem in varied research fields. In this section, we focused on reviewing the studies having the most similarities with our goals; especially the papers investigating the task scheduling problem in cloud based environment.

A stochastic technique that is based on the mechanism of natural selection and genetics has been used by Gu et al. [7] for proposing a scheduling strategy for cloud based environment. Based on the GA, authors proposed a virtual machine load balancing scheme that allows an optimal resource using the cloud data centers. In the same order of ideas, Ge et al. [8] addressed the task scheduling problem with GA by evaluating all jobs in the job queue. The results of their experiments outputted better load balancing. A multi-objective GA have been used by Liu et al. [9] for improving the overall performance of cloud computing. Authors designed a task scheduling model for reducing the system power consumption while improving the profit of service providers by providing a dynamic selection mechanism according to the real time requirements.

Based on the study of cloud computing system structure and its operation mode, Zhu et al. [10] introduced a new resource scheduling strategy based on the ACO algorithm for promoting a new business calculation model in cloud computing environments. In their proposed design, classification of users tasks based on priority in order to ensure requirements of different QoS metrics, is achieved for allocating resources and scheduling tasks. Their proposal showed better makespan compared a random distribution algorithm. However, the proposal does not take into account the variation of task size that is an important target of the scheduling algorithms in cloud environments.

Wei et al. [11] extended the task scheduling problem to the mobile cloud computing environments by extending the Cloudlet architecture. Authors have taken each tasks profit into consideration in order to maximize the profit of the system, which is an import target of the task scheduling algorithm in the commercial mobile cloud environment. They designed their proposal based on the hybrid ACO algorithm, which has been validated by experiments.

Moreover, heterogeneous cloud are usually considered by the cloud providers in order to well manage the utilization of their computing resources. Dai et al. [12] analyzed the structure of heterogeneous cloud, and proposed a framework of multi-objective constrained resource management that extended the computing power and the availability of cloud resources. Then, in order to highlight a trade off among performance, availability, and cost of Big Data application running on Cloud infrastructures, Dai et al. [13] have modeled a mechanism of resource management for heterogeneous clouds by a multi-objective optimization algorithm.

Despite the number of research in this field, the task scheduling in cloud based environments is shown to be NP-hard. Thereby, it is necessary to propose swarm intelligence based algorithm to addressing the task scheduling problem in cloud computing. Furthermore, the process of assigning tasks to available resources that takes into account the characteristics and tasks' constraints is one of the key issues of cloud computing research. Scheduling in cloud based environment is responsible for selecting the most appropriate resources for the execution of given tasks. Before presenting the proposed scheme and algorithm for an optimal scheduling, let us formulate the task scheduling problem.

III. Task scheduling problem formulation

In computer science, scheduling refers to a set of policies and mechanism built into the operating system that govern the order in which the work to be done by a computer system [14]. Given time and resources availability constraints, task scheduling fundamentally refers to a method by which work specified by some means is assigned to resources that have the role of completing the assigned work. In context of cloud computing, scheduling play a vital role since there is always a need of an efficient load balancing in such environments, while maintaining the best system throughput. Therefore, the main aim of task scheduling is to complete jobs' executions while using the minimum amount of resources. However, task scheduling problems differs widely in the nature of the constraints that must be satisfied as well as the mode of desired

schedule. Resources allocation, task scheduling and scaling play key roles in the improvement of reliability and flexibility of systems in cloud-based environment.

Formally, let $T = \{\lambda_1, \lambda_2, ..., \lambda_l\}$ be the set of l tasks $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$ and $M = \{m_1, m_2, ..., m_m\}$ be the set of m virtual machines $m_k,\ k \in [1, m] \cap \mathbb{N}$. Let us consider $J = \{j_1, j_2, ..., j_n\}$ as a set of n jobs $j_j,\ j \in [1, n] \cap \mathbb{N}$ that have to be processed on virtual machines. Each task $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$ is constituted by the a number of jobs of $j_j, \in J$. Throughout, let us assume that each virtual machine $m_k,\ k \in [1, m] \cap \mathbb{N}$ handles at most one job at time. Each job $j_i,\ i \in [1, n] \cap \mathbb{N}$ with size $\sigma(j_i)$ spent a processing time p_j on a given virtual machines $m_k,\ k \in [1, m] \cap \mathbb{N}$ on which it required processing, a release data $\rho(j_i)$ on which the job $j_i,\ i \in [1, n] \cap \mathbb{N}$ becomes available for processing, a due date $\delta(j_i)$ by which the job should ideally be completed and a weight $\omega(j_i)$ indicating the relative priority of the job.

In addition, let $\zeta(\lambda_i)$ be the completion time of the task $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$ and $\zeta(j_j)$ be the completion time of the job $j_j,\ j \in [1, n] \cap \mathbb{N}$. By convention, the sum of completion time of the jobs in a task is the completion time of that task. Also, let $\Pi(\lambda_i)$ be the processing time of the task $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$. Without lost of generality, it value is equal to the longest processing time among the jobs in the task. The task scheduling problem is brought back the a problem of finding the best schedule of jobs in order to minimize the the overall flow time. Hence, each instance of a scheduling problem may consist of three main components namely the set of jobs J known as consumers, the set of virtual machines M known as resources and a set of policy. Thereby, for each task $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$, the optimization problem is modeled in Equation 1 subjected to constraints given in Equations 2-4.

$$\text{Minimize} \sum_{j=1}^{n} \zeta(j_j) \tag{1}$$

Under constraints

$$\sum_{j=1}^{n} \sigma(j_j) \leqslant \tau(\lambda_i) \tag{2}$$

$$\Pi(\lambda_i) \geqslant \sum_{j=1}^{n} p_j, \quad k \in [1, m] \cap \mathbb{N} \tag{3}$$

$$\zeta(\lambda_i) \geqslant \zeta(\lambda_{i-1}) + \Pi(\lambda_i) \tag{4}$$

The constraint given in Equation 2 ensures that the total size of jobs of a task $\lambda_i,\ i \in [1, l] \cap \mathbb{N}$ cannot exceed the virtual machine capacity $\tau(\lambda_i)$ defined by the cost function given in Equation 5. The constraint given in Equation 3 control the processing time of each task. In order to ensure that the completion time $\zeta(\lambda_i)$ of the task λ_i is greater than the completion time $\zeta(\lambda_{i-1})$ of its predecessor task plus its proper processing time $\Pi(\lambda_i)$. We defined the corresponding constraint given in Equation 4. Moreover, without lost of generality, let us assume that $\zeta(\lambda_1) \geqslant \Pi(\lambda_1)$.

Now, let us consider: α as a field specifying the virtual machine environment; β as a field of job's characteristics that include job splitting possibility, known constraints on available resources, order relation among jobs, etc; γ as a field o optimality criterion that may include the completion time as well as the lateness. Then, the combination of job, machine and scheduling behaviors is similar to a 3-field problem classification $\alpha|\beta|\gamma$.

Furthermore, a scheduler can be seen as a program usually operate at operating system level, which has the role of selecting the next job to be admitted into the scheduling system. In cloud computing, a scheduling system constantly performs three steps namely resources discovering and refining, followed by the selection of target resources for given tasks and ended by the association of resources with tasks. Moreover, designing a proper scheduler to satisfy a set of constraints is fundamental to many applications. Indeed, reaching optimal scheduling solution is know to be a NP-complete problem [15]. In most cases, research works on scheduling are focused on sub-optimal scheduling solutions that include heuristic approaches. These methods, rely on rules-of-thumb to guide the scheduling process in the right way reaching a near-optimal solution of the scheduling problem.

IV. Scheduling optimizer: an ACO based case

Recently, artificial intelligence based schemes to solve scheduling problems have been proposed [16], [17], [18], [19], [20]. In cloud computing environment, the ACO is a well known and well-established swarm intelligence based search technique for tackling a wide variety of NP-hard combinatorial problems for optimization purpose. ACO is a stochastic local search method that has been inspired by the pheromone trail laying and following behaviors of some ant species [21]. However, with a large research space, the ACO algorithm become slower and more complex. In addition, a number of realistic problems involve more objective whereas the classical ACO is single objective. Therefore, many the extension of ACO algorithm to handle multiple objective functions have been proposed. MMAS is one of the improved variant of the ACO algorithms that aims at preventing the stagnation of the search by more effectively exploiting the search history [22].

A. Scheme design of the proposed CACO

In order to obtain a rapid convergence time in our proposal and low computing cost, we adopted a modified version of the MMAS, which is an improved version of the ant system. In the design of the proposed scheme, the array of jobs is constructed according to the MMAS algorithm by exploiting the best solutions found during the search. Then, to avoid an early stagnation of the search, the pheromone is set by a minimum value ε_{min} and a maximum value ε_{max}. In our design, in order to kept a consistent list of available resources, a virtual machine $m_k,\ k \in [1, m] \cap \mathbb{N}$ is removed from the list of available resources when its pheromone is less than ε_{min}. The initialization of pheromone of each resource, i.e., the pheromone trail, which is initialized to its maximum value and calculated according to its performance, i.e., the number of processors, the processor amount of instructions per second, the memory size and the rate of flow. The corresponding cost function is given in Equation 5.

The proposed algorithm for task scheduling in cloud-based environment operates in three main steps that include the

initialization step, the construction step and the updating of pheromone trail steps. In the initialization step, a number of parameters such as virtual machines parameters, amount of ants, the maximum number of iterations, the pheromone trail as well as the heuristic information. The pheromone value $\tau_{ik}(t),\ i \in [1,n] \cap \mathbb{N},\ k \in [1,m] \cap \mathbb{N}$ of the virtual machine m_k at iteration t is initialized according to the function given Equation 5. So, the pheromone trail at iteration t is represented by the matrix $\tau_{ik}(t),\ i \in [1,n] \cap \mathbb{N},\ k \in [1,m] \cap \mathbb{N}$. Then, let us consider that the desirability of scheduling the job j_j directly after the job j_i during the iteration t, is equal to $\tau_{ik}(t)$.

$$\tau_{ik}(t) = \frac{1}{\sum_{k=1}^{m} \phi_k} \times \left(\xi_k + \phi_k \cdot \psi_k\right) \tag{5}$$

where $\phi_k,\ \psi_k,$ and ξ_k are respectively the processor million instructions per second (MIPS), the number of processors and the available rate flow on the virtual machine m_k.

Moreover, the heuristic information $\eta_{ik}(t)$ that is seen as the capacity of a virtual machine m_k to process the job j_i is defined in Equation 6.

$$\eta_{ik}(t) = \frac{1}{z(j_i)} \times \left(\upsilon \times \phi_k \cdot \psi_k\right) \tag{6}$$

where $z(j_i)$ is the size of the job j_i, i.e., the number of instructions in the job j_i and υ is a weight factor that should be properly chosen.

Furthermore, the construction step of the proposal is achieved like in the ant system. A colony of ants are independently engaged in the construction of a solution by using the pheromone trails matrix and heuristic information. Concretely, each member of the colony, i.e., ant, will generate an array of jobs that can be considered as a solution and iteratively, build the optimal array of job according the probability function given in Equation 7.

$$p_{ik}(t) = \frac{\tau_{ik}(t) \times \eta_{ik}(t)}{\sum_{i=1}^{n} \left(\omega(j_i) \times \tau_{ik}(t)\right)} \tag{7}$$

Finally, the updating pheromone trails step (see Equation 9) consists in the use of solutions outputted during the construction step in order to improve obtained these solutions. The pheromone trails matrix is updated according to the best ant solution, i.e., the amount of pheromone $\Delta\tau_{ik}$ produced by the best ant, obtained during the construction phase. In our design, the amount of pheromone produced by the best ant is inversely proportional to the flow function f_{ik} (see Equation 8) that allows the scheduling of the job j_i on machine m_k.

$$f_{ik} = \frac{1}{\sum_{i=1}^{n} p_{jk} \cdot z(j_i) + \sum_{k=1}^{m} \phi_k} \times \left(\upsilon \times \phi_k \cdot \psi_k\right) \tag{8}$$

$$\tau_{ik}(t+1) = \rho \times \tau_{ik}(t) + \Delta\tau_{ik} \tag{9}$$

where $0 \leqslant \rho < 1$ is a parameter that express the trail persistence and $\Delta\tau_{ik} = \frac{1}{f_{ik}}$.

B. Description of the proposed algorithm

After the initialization of the pheromone values and the other parameters, each ant of the colony is placed on a resource, i.e., a virtual machine. Tasks submitted for processing in the cloud are aggregated according to their submission times and the corresponding processing and completion time. Then, a set of jobs, i.e., a task or a set of tasks are assigned to ants. Indeed, sets of jobs are associated with one or more ants and each virtual machine is assigned to one or more ants. The scheduling order computed according to models given in Equations 5-7. The algorithm of proposed scheduling task in cloud-based environment is given as follow.

Algorithm 1 . The CACO Algorithm

Begin
Step 1.
1. Initialize parameters
   the amount of ants A_a, the maximum
   number of iterations t_m, the weight factor υ
   and the trail persistence ρ
2. Initialize pheromone trail matrix τ_{ik}
   according to Equation 5
3. Build the set of tasks
4. Build set of jobs
5. Associate each set of jobs with an ant

Step 2.
5. Initialize the iteration index ($t = 1$)
   Repeat
   For each ant **do**
6. Proceed to jobs scheduling according to
   models given in Equations 5-7
7. Proceed to the local search process
   End for
8. Compute the flow function f_{ik}
9. Proceed to pheromone trails update
   according to Equation 9
10. Increment the iteration index ($t = t + 1$)
   Until ($t > t_m$)
End

V. Performance evaluation

Series of experiments have been conducted on the well-established simulated cloud computing environment provided by CloudSim [23]. CloudSim is embedded with a generalized and scalable framework that allows modeling and simulation of a virtualized cloud computing infrastructures and application services. The speed of each processing element is expressed in MIPS and each cloudlet (task) length is expressed as the number of instructions to be executed. We adopted a scenario of a homogeneous cloud computing environment for our experiments. The capacity of machines is assumed to be the same in all experiments. The performance of the proposed CACO algorithm is evaluated by comparing its results with the GA based task scheduler for the cloud computing systems proposed by Ge and Wei [8], and with the ACO based scheduling task in cloud computing environment proposed by Zhu et al. [10].

A. Simulation environment and parameters

The simulation environment consists of a data center of 4 identical physical Intel Xeon Quad Core server with 8GBytes of random access memory, 2 $\times$ 500GBytes SATA hard drives and CentOS 6.0 operating system. In each physical machine, 10 homogeneous virtual machines of equal capacities that includes 2 processors and 2GBytes of random access memory, are deployed. A shared memory policy in which resources are shared between physical an virtual machine is adopted for the scheduling.

Experiments was carried out with a number of randomly generated jobs (cloudlets) ranges from 20 to 80 in series of 20. The number of instructions in each jobs varies from 500 to 1500 and from 1500 to 2000. So, two tests scenario have been considered. Each job handles a data size of 300 generated from a discrete uniform distribution and also requires only one processor for its execution. The processing time p_j of a job is drawn from a uniform random distribution with limit $[1,\ 100]$.

B. Results and discussion

In the two scenario of experiments, we measured the execution time of the series of jobs. In each scenario, we proceeded to 10 runs of the proposed CACO as well as GA and ACO solutions in the same test conditions. In each test scenario, results of the proposed CACO are compared to results obtained in the aforementioned compared solutions. The obtained results from the CACO and the compared scheduling solutions are reported in Tables I and II. In these tables, the average makespan and the average relative standard deviation of solutions are reported in each scenario, i.e., in each size of jobs.

TABLE I: Average makespan ($\bar{\mu}$) and standard deviation (σ) when the number of instructions in each jobs varies from 500 to 1500

Job size	*Scheduling solutions*					
	CACO		**ACO**		**GA**	
	$\bar{\mu}$	σ	$\bar{\mu}$	σ	$\bar{\mu}$	σ
20	31.23	*0.00*	36.93	0.31	43.11	*0.52*
40	121.69	*0.09*	135.63	*0.36*	141.69	*0.54*
60	301.83	*0.53*	319.91	*0.46*	336.93	*0.55*
80	569.06	*0.57*	577.31	*0.51*	602.13	*0.58*

TABLE II: Average makespan ($\bar{\mu}$) and standard deviation (σ) when the number of instructions in each jobs varies from 1500 to 2000

Job size	*Scheduling solutions*					
	CACO		**ACO**		**GA**	
	$\bar{\mu}$	σ	$\bar{\mu}$	σ	$\bar{\mu}$	σ
20	62.09	*0.00*	62.83	0.32	70.01	*0.6!*
40	220.13	*0.08*	232.16	*0.39*	235.91	*0.64*
60	505.09	*0.57*	525.13	*0.48*	539.13	*0.66*
80	937.19	*0.61*	949.01	*0.53*	965.12	*0.67*

Figure 1 presents the results of execution time, i.e., the makespan, when the number of instructions in each jobs varies from 500 to 1500 while Figure 2 highlights the execution time when the number of instructions in each jobs varies from 1500 to 2000. As it can be observed from Figures 1 and 2, the overall performance of the CACO algorithm is better than that of the ACO and GA in all tests scenario, which reveals the effectiveness and efficiency of the proposed scheme. This gain is mainly favored by the adopted features of MMAS and by the proposed models (see section IV-B) for jobs scheduling.

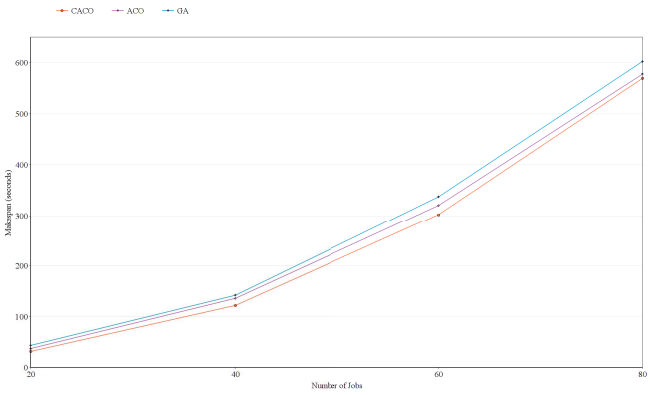

Fig. 1: Comparison of the solution quality when the number of instructions in each jobs varies from 500 to 1500.

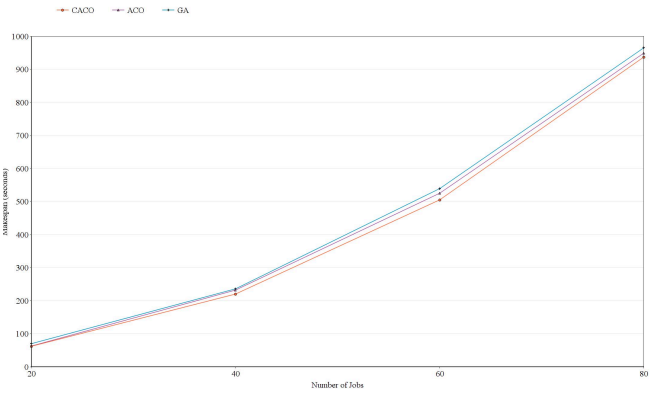

Fig. 2: Comparison of the solution quality when the number of instructions in each jobs varies from 1500 to 2000.

However, from average relative standard deviations reported in Tables I and II, we must recognize that the standard deviation in the proposed CACO increases when the size of jobs increases (job size of 60 and 80) compared to the standard deviation of the ACO solutions. Unfortunately, the GA solution outputted high makespan and a lot of dispersion in results. This due to the genome encoding scheme adopted by Ge and Wei [8].

VI. Conclusion

In this paper, we considered the task scheduling problem in cloud based environment, which is a challenging issue that

significantly affects the performance of cloud data centers. We proposed an energy aware scheme for task scheduling by using some features of the ant system. The major contributions of our proposal are illustrated as follows. Firstly, we formulated the problem task scheduling in the cloud based environment. Secondly, we proposed some models and an algorithm based on the ant system for scheduling jobs, which minimize the task execution time and efficiently use cloud resources. Thirdly, simulation of the proposal was made. To evaluate the effectiveness of the proposed algorithm, a large number of tests have been performed using randomly generated test instances. The obtained results show that, the proposal delivers interesting performance, thereby demonstrates it effectiveness. As future work, we plan to introduce linear programming based models for optimizing the local search process and also, we planed to evaluate metrics like high throughput and low response time. An interesting extension will be the consideration of jobs due date related in the optimization objective. More biologically inspired solutions like the Artificial Bee Colony and the Particle Swarm Optimization may be applied for solving the task scheduling problem in cloud environments and therefore, it is an opportunity for the future research.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] L. Chen, M. Qiu, J. Song, Z. Xiong, and H. Hassan, "E2FS: an elastic storage system for cloud computing," *The Journal of Supercomputing*, pp. 1–16, 2016.

[3] L. Chen, M. Qiu, W. Dai, and H. Hassan, "An efficient cloud storage system for tele-health services," *The Journal of Supercomputing*, pp. 1–17, 2017.

[4] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[5] A. A. A. Ari, B. O. Yenké, N. Labraoui, I. Damakoa, and A. Gueroui, "A power efficient cluster-based routing algorithm for wireless sensor networks: Honeybees swarm intelligence based approach," *Journal of Network and Computer Applications*, vol. 69, pp. 77–97, 2016.

[6] A. A. A. Ari, A. Gueroui, N. Labraoui, B. O. Yenké, C. Titouna, and I. Damakoa, "Adaptive scheme for collaborative mobile sensing in wireless sensor networks: Bacterial foraging optimization approach," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on*. IEEE, 2016, pp. 1–6.

[7] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," *Journal of Computers*, vol. 7, no. 1, pp. 42–52, 2012.

[8] Y. Ge and G. Wei, "GA-based task scheduler for the cloud computing systems," in *Web Information Systems and Mining (WISM), 2010 International Conference on*, vol. 2. IEEE, 2010, pp. 181–186.

[9] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li, "Job scheduling model for cloud computing based on multi-objective genetic algorithm," *IJCSI International Journal of Computer Science Issues*, vol. 10, no. 1, pp. 134–139, 2013.

[10] L. Zhu, Q. Li, and L. He, "Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 54–58, 2012.

[11] X. Wei, J. Fan, Z. Lu, and K. Ding, "Application scheduling in mobile cloud computing with load balancing," *Journal of Applied Mathematics*, vol. 2013, 2013.

[12] W. Dai, H. Chen, and W. Wang, "Rahec: A mechanism of resource management for heterogeneous clouds," in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*. IEEE, 2015, pp. 40–45.

[13] W. Dai, L. Qiu, A. Wu, and M. Qiu, "Cloud infrastructure resource allocation for big data applications," *IEEE Transactions on Big Data*, 2016.

[14] M. Bagaa, Y. Challal, A. Ksentini, A. Derhab, and N. Badache, "Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1339–1368, 2014.

[15] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on software engineering*, vol. 14, no. 2, pp. 141–154, 1988.

[16] M. Qiu, Z. Ming, J. Li, K. Gai, and Z. Zong, "Phase-change memory optimization for green cloud with genetic algorithm," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3528–3540, 2015.

[17] R. Jena, "Task scheduling in cloud environment: A multi-objective ABC framework," *Journal of Information and Optimization Sciences*, vol. 38, no. 1, pp. 1–19, 2017.

[18] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm," *Journal of Parallel and Distributed Computing*, vol. 101, pp. 41–50, 2017.

[19] S. Xu, Y. Liu, and M. Chen, "Optimisation of partial collaborative transportation scheduling in supply chain management with 3PL using ACO," *Expert Systems with Applications*, vol. 71, pp. 173–191, 2017.

[20] A. A. A. Ari, I. Damakoa, A. Gueroui, C. Titouna, N. Labraoui, G. Kaladzavi, and B. O. Yenké, "Bacterial foraging optimization scheme for mobile sensing in wireless sensor networks," *International Journal of Wireless Information Networks*, vol. 59, no. 3, pp. 254–267, 2017.

[21] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[22] T. Stützle and H. H. Hoos, "MAX–MIN ant system," *Future generation computer systems*, vol. 16, no. 8, pp. 889–914, 2000.

[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.