Journal of King Saud University – Computer and Information Sciences (2016) **28**, 55–67

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

A novel algorithm for reducing energy-consumption in cloud computing environment: Web service computing approach

N. Moganarangan [a], **R.G. Babukarthik** [b,*], **S. Bhuvaneswari** [b], **M.S. Saleem Basha** [b], **P. Dhavachelvan** [b]

[a] *Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tamil Nadu, India*
[b] *Department of Computer Science, Pondicherry University, Puducherry, India*

Received 11 October 2013; revised 8 March 2014; accepted 3 April 2014
Available online 17 November 2015

KEYWORDS

ACO ant colony optimization;
CS cuckoo search;
VSF voltage scaling factor;
EcPSO extended compact particle swarm optimization

Abstract Cloud computing slowly gained an important role in scientific application, on-demand facility of virtualized resources is provided as a service with the help of virtualization without any additional waiting time. Energy consumption is reduced for job scheduling problems based on makespan constraint which in turn leads to significant decrease in the energy cost. Additionally, there is an increase in complexity for scheduling problems mainly because the application is not based on makespan constraint. In this paper we propose a new Hybrid algorithm combining the benefits of ACO and cuckoo search algorithm. It is focused on the voltage scaling factor for reduction of energy consumption. Performance of the Hybrid algorithm is considerably increased from 45 tasks onward when compared to ACO. Energy consumed by Hybrid algorithm is measured and energy improvement is evaluated up to 35 tasks. Energy consumption is the same as ACO algorithm because as the number of tasks increases (45 to 70) there is a considerable decrease in the energy consumption rate. Makespan of Hybrid algorithm based on number of tasks is compared with ACO algorithm. Further we have analyzed the energy consumption for a number of processors and its improvement rate – up to 6 processors, energy consumption is considerably reduced and the energy consumption tends to be in steady state with further increase in the number of processors.

* Corresponding author.
E-mail addresses: rengannsj77@gmail.com (N. Moganarangan), r.g.babukarthik@gmail.com (R.G. Babukarthik), booni_67@yahoo.co.in (S. Bhuvaneswari), m.s.saleembasha@gmail.com (M.S. Saleem Basha), dhavachelvan@gmail.com (P. Dhavachelvan).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2014.04.007

1. Introduction

Cloud computing is becoming one of the predominant approaches in rendering IT services by reducing cost for the consumers. The approach not only influenced techniques used in computing but in turn processes, technology used for constructing and managing IT within the service provider and the enterprise. By offering a secure computing paradigm, cloud computing is becoming an important platform for scientific application. On-demand facility of virtualized resources as service is offered using virtualization in cloud computing without any delay (Venkatesan et al., 2013; Rajeswari et al., 2014).

Cloud computing technologies offer major benefits to the IT industries such as elasticity and rapid provisioning which includes increasing or decreasing the infrastructure facilities for a particular time based upon the required needs. Pay-as-you-go-model deals with the organization that requires any services and pay for the exact amount of resources they utilized in terms of infrastructure, platform and software as services. Capital cost is reduced as organizations do not need to have an inbuilt infrastructure, thereby resulting in the reduction of infrastructure. Access to unlimited resources in cloud computing means that the cloud provider has been able to deploy hundreds of server instances simultaneously; thereby it is possible to access unlimited resources. Flexibility means that deploying cloud instances by means of varying hardware configuration, various operating systems and different software packages (Dhavachelvan et al., 2006; Dhavachelvan and Uma, 2005). Some benefits of the cloud include *fault tolerance and high availability*. Since the cluster worker nodes are spread around the cloud sites, in the event of cloud down time or failure, cluster operations will not be interrupted at any cost of time as the worker nodes will take care of it. *Infrastructure cost reduction*: the pricing models among the cloud providers may vary considerably; the cluster node will change the location from one provider to another one thus reducing the overall infrastructure cost.

The main reason behind focusing on energy efficiency is due to the increase in energy cost spent on data center. The server machine is the vital component for increase in electrical cost. The electrical cost of the server machine is due to direct power consumption and cooling equipment used in it. In the data center 75% of energy cost is due to direct power consumed by server machine and indirect power used for cooling equipment. Additionally, due to the use of high performance multi-core processors in server machine, there exists power hunger and dissipation of considerable heat.

The following work is contributed:

- Proposal of a new Hybrid algorithm using ACO and cuckoo search.
- Analysis of job creation time, task creation time, destruction time, result retrieval time and total time for Hybrid algorithm.
- Performance comparison of a new Hybrid algorithm and ACO algorithm.
- Makespan improvement comparison of a new Hybrid algorithm with ACO algorithm.
- Energy comparison of a new Hybrid algorithm and ACO algorithm.
- Studied energy and makespan based on the number of tasks for Hybrid algorithm.

The remainder of this paper is structured as follows: Section 2 deals with previous work that has been carried for scheduling job in cloud computing for minimization of energy, makespan and resources. In Section 3 we propose a new Hybrid algorithm for scheduling job using ACO and cuckoo search, Procedure for Hybrid algorithm and pictorial representation of Hybrid algorithm using flow charts. Section 4 describes the implementation details of Hybrid algorithm and its performance, energy, makespan which has been compared with ACO algorithm. Section 5 states the conclusion and direction for future research.

2. Related work

Parallel bi-objective genetic algorithm is based on Energy-conscious scheduling heuristic. It minimizes the energy consumption and the make span. The major drawback is that it consumes more resources (Mezmaz et al., 2011). Without detailed information of participating node or centralized node, Community-Aware Scheduling Algorithm (CASA) increases both average job waiting time and job slowdown radically (Huang et al., 2013). Elastic cluster architecture supports execution of heterogeneous application domain, which dynamically partitions cluster capacity and adapts to variable demands (Montero et al., 2011). Performance of cloud computing services is analyzed for scientific computing workloads based on loosely coupled applications (Iosup, 2011). Based on network-flow-theory is modeled an algorithm for data center to reduce energy and virtual machines migration thereby reducing the overhead of virtual machines (SiYuan, 2013). For achieving optimal growth in various cloud infrastructure mathematical models are proposed stating that the response time of the slowest nodes is not more than three times of fastest node (Yeo and Lee, 2011). The algorithms depict how to achieve predictability and feasibility (Duan et al., 2007). On the basis of the Berger model, job scheduling algorithm is proposed, generally user tasks is classified by the model based on resource fairness justice function and QoS preferences to judge fairness of resource allocation (Xu et al., 2011). Across various multiple data centers near-optimal scheduling policies are achieved by cloud provider based on factors of energy efficiency such as carbon emission rate, energy cost, CPU power efficiency, and workload, (Garg et al., 2011). In case of dynamic-urgent cloud environment a good support is provided by layered and historical queuing performance model. It provides guidelines for parameterizing the models at a lower overhead (Bacigalupo et al., 2011). The workload that measured hybrid configuration compared to local setup reveals performance-cost ratio from analyzing the cost of multi-cloud (Moreno-Vozmediano et al., 2011). Gross cost is reduced in life time of entire application in elastic cloud computing by determining optimal number of computing resources per charge unit using partitioned balanced time scheduling (Byun et al., 2011). Inter-arrival time, status, parallel runtime, user, request time and application are features of failed job (YulaiYuan, 2012). Traditional formulation of scheduling problem is covered by algorithm such as trust dynamic level scheduling, for enabling cloud environment execution time

and reliability of applications is considered simultaneously. It reduces failure probability of task assignments and assured secured environment execution of tasks (Wang et al., 2012). For job shop scheduling problem it states the approach on basis of ant colony optimization (ACO) and particle swarm optimization (PSO). Every machine is provided with an objective to find possible solution to reduce waiting time and completion time (Sumathi, 2010).

Heuristic of ant colony optimization (ACO) states clearly for given model of target architecture and applications, it executes mapping and scheduling and for optimizing application performance. Exploring various solutions for mapping and scheduling tasks execution time is reduced by using ACO. Maintaining the best correlation among problems and reduction of execution time of exploration is carried out by multi stage decision process (Ferrandi et al., 2010). Analysis of fault recovery and grid service reliability modeling is studied in grid systems using Local Node Fault Recovery (LNFR) mechanism. Its' main use is that it allows life time for number of recovery carried and grid sub task, exact fault recovery strategies based on local situations is chosen by resource provider. The drawback is that link and node satisfy poison processes, hence in all cases it is not true. (Guo et al., 2011). The scheduling model consists of two agents and set of processing machine whose jobs sizes are not alike is taken into consideration. Pareto optimal solutions are derived by using improved ACO algorithm. Makespan is reduced using two agents; in batch processing priority is given for jobs from same agent. To select next jobs to add in the current batch processing, state transition probability is used (Tan et al., 2011). Particle swarm distribution algorithm is estimated using novel framework. Applying selection to local best solutions, optimal solution is obtained. From selected solutions probabilistic model is constructed. From PSO particle moving mechanism and EDA's model sampling method a new individuals are created by stochastic combination. Combining advantage of extended compact genetic algorithm with binary PSO developed extended compact particle swarm optimization (Tavakkoli-Moghaddam et al., 2011). Based on evolutionary algorithm and fuzzy system improved Wang-Mendel model based on PSO is proposed. Modified particle swarm optimization algorithm is adopted for optimizing fuzzy rule, extrapolating complete fuzzy rule is derived (Ahn et al., 2010). Intelligent Dynamic Swarm uses Rough Set theory and feature selection based on PSO, vagueness and uncertainty are handled by a mathematical tool using K-means algorithm (Yang et al., 2010). For multi-objective job scheduling problem a new PSO algorithm is created to solve unceasing optimization problems. Particle position representation, particle velocity and particle movement are modified to solve scheduling problems of discrete solution space (Bae et al., 2010). Grid workflow trustworthy scheduling is solved using rotary chaotic particle swarm optimization. Scheduling performance is optimized in multi-dimensional complex space. Some optimization methods are canceling history velocity, detecting precise time, double perturbation of gBest and pBest, and dimension of double perturbation is proposed thereby helping particles to escape from local optimum (Sha and Lin, 2010). Continuous optimization problems can be solved by focusing on endless variable sampling act and hence it acts as key extending ACO for transforming discrete to continuous optimization. SamACO algorithm uses candidate variable for selection, pheromone cooperation and Ant solution constructed (Tao et al., 2011). Convergence and qualities solution is improved using local search procedure on the basis of neighborhood of JSSP (Hu et al., 2010). Novel framework is proposed on the basis of receding horizon control using ant colony system for solving it (Zhang et al., 2010).

Several methods are proposed for reduction of energy consumption using various parameters, but a very few concentrated on reduction of energy consumption using scheduling algorithms. Using scheduling algorithms energy can be reduced dramatically only if jobs are scheduled within the allocated time interval. Moreover jobs need to be scheduled within the available resources so that jobs need not to be waiting for resources n number of times. A new scheduling algorithm is proposed for reduction of energy consumption and completion time, where resources are allocated to the jobs with the given time interval.

3. Problem modeling

This section describes the application model, energy model and makespan model.

3.1. Application model

Using direct acyclic graph parallel programs are represented in Fig. 1. Graph G = (n, e) contains a set of 'n' nodes and 'e' edges. A Task graph is one in which nodes denote tasks and it is partitioned from an application and the preference constraint is denoted by edges. An edge $(i, j) \in e$ between the task n_i and task n_j denotes inter-task communication. *Entry task:* it is a task which does not contain any predecessor's n_{entry}. *Exit task:* a task which does not have any successor's n_{exit}. *Most significant parent:* within the predecessor of task n_i, predecessor which completes its communication at modern time is called as Most Significant Parent (MSP). It is denoted by MSP n_i. *Critical path:* the longest path in a task graph is called critical path. *Insertion of task:* between two consecutive tasks which is already assigned to processor if there exists an ideal time slot and a new task can be added to the scheduler such that there is no violation of constraint. By doing so, insertion schemes will try to increase processor utilization time.

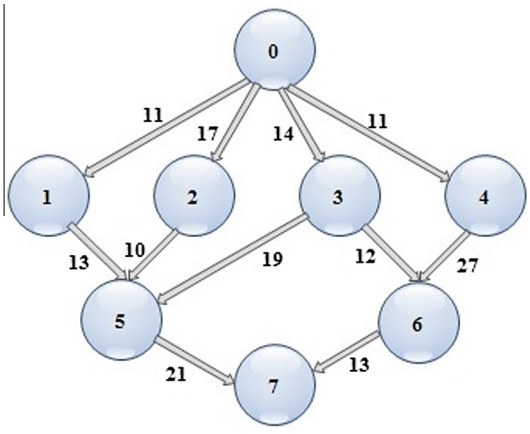

Figure 1 Task graph.

3.2. Energy model

The energy model is a derivative of the power consumption model from complementary metal oxide semi-conductor of logic circuits. Microprocessor based on CMOS is defined as sum of the leakage power short-circuit and capacitive power. Output voltage, input rise time, input voltage level, output loading, and power-dissipation capacitance are the factors affecting power consumption in CMOS.

CMOS Power consumption states are,

(1) Static power consumption.
(2) Dynamic power consumption.

Static power consumption: power consumption in CMOS generally occurs whenever all the input is detained at a certain valid logic level, hence circuit is not in charging states because of this low static power consumption is carried by CMOS devices. This is due to consequences of leakage current. Static power consumption is a product of leakage current L_c and supply voltage S_v (Mezmaz et al., 2011).

$$P_s = \sum (L_c * S_v) \tag{1}$$

Dynamic power consumption: it takes place whenever CMOS requires shifting to high frequency. Dynamic power consumption P_d donates the overall power consumption and it is the sum of transient power consumption P_t and capacitive-load power consumption P_{cl} (Mezmaz et al., 2011).

$$P_d = (P_t + P_{cl}) \tag{2}$$

Transient power consumption P_t is stated whenever current flows when transistors devices are switching from logic state to another state. It leads to current required for charging internal nodes (switching current) and current flowing from VCC to GND. P_t is calculated using equation (Mezmaz et al., 2011):

$$P_t = (C_{pd} * V_{cc}^2 * f_i * N_{sw}) \tag{3}$$

Transient power consumption P_t, V_{cc} represents supply voltage, f_i is frequency input signal, N_{sw} denotes number of bits to be switched, dynamic power-dissipation C_{pd} of capacitance. If the output has the same load and at same output frequency if they are switching then *Capacitive load power consumption* P_{cl} can be estimated using the following equation (Mezmaz et al., 2011),

$$P_{cl} = (C_l * V_{cc}^2 * f_0 * N_{sw}) \tag{4}$$

where, capacitive load power consumption P_{cl}, f_0 is signal frequency output, C_l is load capacitance, number of outputs switching N_{sw}. Energy consumption of application during parallel execution is stated by Mezmaz et al. (2011),

$$E_{app} = \sum_{i=1}^{n} (ACv_i^2 f * w_i) \tag{5}$$

$$E_{app} = \sum_{i=1}^{n} (av_i^2 * w_i) \tag{6}$$

n_i states number of tasks that needs to be executed, supply voltage v_i, w_i is total time taken for n_i task to execute.

3.2.1. Heuristic for energy-conscious scheduling

Whenever the energy consumption in task scheduling is considered the complexity of the problem increases dramatically. Applications are not based on the deadline-constrains. Hence focusing on energy consumption, task scheduling needs more attention and it has to be calculated on the basis of quality schedule. A new heuristic for energy conscious scheduling is relative superiority (RS). For ready task relative superiority value for all processors is evaluated using processor and supply voltage of task. The maximum value of RS is attained based upon the processor. Thus it is clear that energy conscious scheduling decision is confined to local optimum.

3.2.2. Machine energy estimation

Cores are wrapped in processors and processors are grouped in a computing machine while estimating machine energy. Each core is accomplished with active voltage frequency scaling (VFS). Based on supply voltage each core can function with varying speed on the basis of performances (clock frequencies). VFS tend to exploit U-shaped relationship among core supply voltage (speed of execution) and energy consumption. Various cores belonging to same processor are expected to operate at various voltage/frequency points. Generally applications need to specify the voltage/frequency when considering the Energy-efficiency using VFS. The assumption of environment in cloud computing cannot be made because of GNU/Linux management tools of kernel power. Kernel version 2.6.35 – 25 tells that VFS is dynamic and self-regulated. A sampling rate of 10 ms (time period of VFS change) is the default value for on-demand governor. Instructions are on the basis of CPU utilization when there is a local decision making based on a global one and it is controlled by kernel. In this case, using '*cpu-freq*' tools power is coped by operating systems in on-demand governor. Since on-demand governor implemented '*race-to-idle*' policy voltage/frequency is fixed to maximum value whenever CPU is needed. When CPU utilization decreases radically, based on load voltage/frequency is chosen. A jitter will occur whenever there exists a spontaneous adjustment of CPU frequency, in large-scale system this will in turn cause delayed communication. There exist several components apart from processor consuming energy. Our main aim is to focus on total energy consumption and there remain various components that need to be included. Power model is specified by a relation (Mezmaz et al., 2011),

$$P_m = (P_{const} + P_{high}) \tag{7}$$

where, P_m is total power of machine, P_{const} is power constant, P_{high} is max power for core machine. Energy is considered to be the product of power and time, in some cases energy replicates to '*race-to-idle*' policy.

Total energy can be stated as sum of maximum time taken by the core machine and the completion time of tasks assigned to all core machines and its power (Mezmaz et al., 2011).

$$E_{mach} = \left(P_{const} + C_{tmax} + P_{high} * \sum_{c}^{cores} C_{tc} \right) \tag{8}$$

E_{mach} is the energy of machines, c_{tc} states completion time of task assigned to core machines, C_{tmax} denotes maximum time of core machine. If currently no task is running then the machine is considered to be switched off (see Table 1).

Table 1 Energy parameters.

Sl. No.	Index	Value range
Energy parameters		
1	Completion time	15–30 [time unit]
2	Maximum rate Contention	30–60%
3	Time limit	0–30%
4	Power constant	15–25 W
5	High Power	0.1–2.0 W
6	A	0.7–1.2

3.3. Makespan model

Job scheduling is a combinatorial optimization problem in the field of computer science, where the ideal jobs are assigned to the required resource at a particular instant of time. The description is as follows. *Makespan or completion time* is the total time taken to process a set of jobs for its complete execution. Minimization of makespan can be done by assigning the set of J_i jobs to set of virtual machines vm, the order of execution of the jobs in virtual machines does not matter.

3.3.1. Notations

Let J_i represent the job and P_j denotes the processing time of jobs, and thus processing time of job set B (Mezmaz et al., 2011), can be defined as

$$P(B) = \sum J_{i\in B} * P_j \tag{9}$$

If π is a possible schedule for a given scheduling problem, S_j is the starting time of job J_i in a possible schedule. E_j Denotes the end time of job J_i, P_j is the processing completion time of job J_i (Mezmaz et al., 2011),

$$P_j = \sum (E_j - S_j) \tag{10}$$

N_j denotes number of jobs, C_j is the completion time of job J_i. Let J_i be the set of jobs (J_1, J_2, J_3, … J_i) that need to process and π be the possible schedule for a given job scheduling problem. J_i of jobs need to processed by the virtual machine vm_m = (vm_1, vm_2, vm_3, … vm_m). Where 'm' is the mth virtual machine, the minimal value of the makespan (completion time) among all the possible schedules is given by the processing time of the operations P_j = (P_1, P_2, P_3, … P_j), C_{max} denotes the completion time.

3.3.2. m parallel virtual machines scheduling problem

Let us considered that 'm' parallel virtual machines is available, and now at time being let us assume that one is always in a busy state and it is not available for the job execution. To perform the scheduling jobs are arriving J_i = (J_1, J_2, J_3, … J_n) and it is necessary to schedule the jobs to available virtual machines. Constraints are that new jobs that need to be scheduled arrive only after already existing jobs are scheduled. Let us assume that virtual machine vm_1 is periodically unavailable and the virtual machine which is not available will start at the unavailable period of time. The aim is to minimize the makespan (completion time). For our assumption, let the length of the available virtual machine and unavailable virtual machine be 1 and $\alpha > 0$, respectively. p_m, denotes the processing time of the virtual machines, c_{max} is the completion time c_{online} and c_{offlin} algorithm respectively.

For a given problem p_m, $vm_1|online|c_{max}$ and thus, there is no online algorithm with lower bound of ratio less than 2.

Let β be positive number of small value, and jobs J_i = (J_1, J_2, J_3, … J_i arriving have a common processing time as β.

Case 1. It is possible that one virtual machine can process two jobs that are jobs J_1 and J_2.

In such a scenario $c_{online} \geqslant 2\beta$, but in the offline schedule each virtual machine will be processing one job at a time that is $c_{offline} = \beta$.

$$\frac{C_{online}}{C_{offline}} \geqslant \frac{2\beta}{\beta} \tag{11}$$

Cancel the both numerator and denominator.

$$\frac{C_{online}}{C_{offline}} \geqslant 2 \tag{12}$$

Case 2. If all the virtual machines process one job at a time

After completion of first set of schedule, if we provide the second set of job to be scheduled to the vm for the job J_{vm+1}-…J_{2vm} with processing time 1.

$$C_{online} \geqslant \min(\beta + 2, 2 + \alpha) \tag{13}$$

Likewise in the offline scheduling algorithm each and every virtual machine is capable of processing one vm only. If the second set of jobs is given to the virtual machine then vm_2-…vm_m, schedules.

$$C_{offline} = (1 + 2\beta) \tag{14}$$

and evaluating the equations

$$\frac{C_{online} \geqslant \min(\beta + 2, 2 + \alpha)}{C_{offline} = (1 + 2\beta)} \tag{15}$$

and then on evaluating further,

$$\frac{C_{online}}{C_{offline}} \to 2 \tag{16}$$

$$\beta \to 0 \tag{17}$$

Hence for a given problem p_m, $vm_1|online|c_{max}$ and thus, there is no online algorithm with lower bound of ratio less than 2.

4. Hybrid algorithm

Combining advantage of ant colony optimization and cuckoo search, a new Hybrid algorithm has been developed for combinatorial optimization problems. The disadvantage of ant colony optimization has been overcome by cuckoo search that is in ant colony optimization ant moves in random directions for search of food source around the colony. Chemical substances named pheromone is deposited on the path. While solving optimization problems it traps the ants

and hence to perform local search time taken is considerably more. The above draw backs can be overcome by using cuckoo search. Cuckoo search is used to perform local search in ant colony optimization. The major advantage of using cuckoo search is that, distinct parameter is used apart from population size.

4.1. Description of ACO and cuckoo search

4.1.1. Ant colony optimization

For solving computational problems ant colony optimization technique can be used because of the probabilistic nature, ant colony optimization is used to discover best path through graphs, based on activity of ants looking for a path among their colony in search of food source. This idea has been used to solve various numerical problems; many problems have come out based on various distinct features of ant behaviors.

Explanations: ant moves in random directions in search of food source around the colony, if food source has been discovered by ant it will come directly to nest, leaving a trail of pheromone in path. Since pheromone is attractive by its nature the rest of ants tend to follow directly along that path. Once coming back to their colony they further leave a trail of pheromone in path, which will in turn strengthen the route. If there exist more than one route to reach an identical food source, shorter path will be traveled by many number of ants, than longer path because pheromone deposited in the longer path will be evaporated for a particular instant of time. This is due to the volatile nature of pheromones, finally all ants decided to travel shortest path. Generally environment is used as a communication medium by ants, for exchange of information among ants and it takes place with the help of pheromone that has been deposited. The scope of information exchange is local, those colonies where ant located pheromones left has a belief for them.

Local decision policies (trails and attractiveness): In ant colony algorithm ants try to construct a solution for a given problem iteratively, once the solution is constructed for the problem. Evaluation of solution will be carried out by ant and then will try to modify trail value which is used in construction of solution. The modified pheromone information is used by future ants to search further.

Trail evaporation and daemon: reduction of trail value will be carried out by trail evaporation to avoid getting stuck further in local optima. Searching of non-local perspective is carried out by daemon.

Edge selection: in ant colony optimization algorithm ant acts as a computational agent. It incrementally tries to build solution for the problem, solutions which are derived instantly are called as solution states. Each and every looping of algorithm is considered based on ant movement from state 'm' to state 'n', resulting in a more feasible solution. Each ant 'k' works out a set $A_k(m)$ of feasible elaboration to its current state in each looping, the probability P^k_{mn} of moving from state 'm' to state 'n' is based on arrangement of two values. Attractiveness β_{mn} of move, and trail T_{mn} of move, shows how capable in the past for a particular move. Thus kth ant moves from state 'm' to state 'n' with probability (Ferrandi et al., 2010).

$$P^k_{mn} = \frac{(T^\alpha_{mn})(\beta^\eta_{mn})}{\sum(T^\alpha_{mn})(\beta^\eta_{mn})} \tag{18}$$

T_{mn} denotes the pheromone amount deposited from state 'm' to 'n', 'α' is used for controlling the influence of T_{mn}. β_{mn} is the state transition desirability from 'm' to 'n'. η is used to control influence of β_{mn}.

Pheromone update: if solution is completed by all ants, updated trail equation is (Ferrandi et al., 2010)

$$\beta^\eta_{mn} \leftarrow (1-\rho)\beta^\eta_{mn} + \sum \Delta\beta^k_{mn} \tag{19}$$

where, β_{mn} denotes pheromone amount that has been dumped for state transition mn. ρ is the coefficient for pheromone evaporation. $\Delta\beta^k_{mn}$ states the amount of pheromone dumped by 'kth' ant.

4.1.2. Cuckoo search

Cuckoo search is used for the optimization problem, it has been seen that performance of the cuckoo search is higher than other Meta heuristic algorithms.

Representation of cuckoo search (CS): each and every egg in the nest denotes a solution; a new solution is represented by a cuckoo egg. The main motivation of cuckoo egg is to derive the best solution and to replace the solution, which is not so-good in the nests. Each nest contains exactly one egg. Cuckoo search is based on following rules,

- All cuckoos lays only a single egg at a specific period of time and the eggs are dumped by randomly choosing the nest.
- For the next generation, high quality of eggs in best nest is carried out.
- Generally hosts nests are fixed, the probability of laid egg by cuckoo bird is found by the host bird $p_a \in (0,1)$. On finding this we can further do some operations on worst nests, solution which is derived is dumped for further calculations.

Random walk: the major issues in application of random walk and Levy flights for deriving the new solution is

$$Z_{t+1} = (Z_t + sE_t) \tag{20}$$

E_t, is obtained from Levy flights, or by normal distribution, it is also possible to link similarity between hosts egg and cuckoo egg while implantation will be somewhat tricky. 's' denotes for a fixed number of iterations, how much distant a random walker can go.

If 's' is too large, from the old solution a new solution derived will be far away. In such case, it is necessary to accept. If 's' is too small, the changes are also considerably small and hence search is not efficient. Hence it is more important to maintain a proper step size.

4.3. Hybrid algorithm

Hybrid algorithm	
Step 1:	***Initialization of parameters***
	Set the beginning of pheromone trail, heuristic information (h_{if}), *random nests* (r_{ns})
Step 2:	*Get Input jobs from 1 to n jobs*
Step 3:	***Apply transition rules***
Step 4:	*for each jobs* j_i *to* j_n *do*
	for each virtual machine vm_1 *to* vm_m *do*
	Assign jobs to vm
	$Vm = job\ j_i$
	end for
	end for
Step 5:	***random walk by cuckoo search from Eq.*** (20)
Step 6:	*Pheromone updation*
	for each pheromone p_m *to* p_n *do*
Step 7:	*evaluate* P^{k}_{mn} *using Eq.* (18)
	end for
Step 8:	pheromone trails updation
	for pheromone dumped β_m *to* β_n *do*
	evaluate β_{mn} *using Eq.* (19)
	end for
Step 9:	*If current_jobs* $\geqslant$ *n jobs then*
	n jobs $++$ *and go to step 2*
	else
Step 10:	*Return value*

Algorithm description: parameter initialization such as pheromone trail, heuristic information, and random nest. The jobs are processed based on arrival from 1 to *n* jobs. After applying transition rules, jobs which have arrived need to be processed. For processing of jobs, jobs are assigned to virtual machines based upon the arrival. Thus *n* jobs are assigned to the vm_m virtual machines. Process random walk, for performing the local search, cuckoo search is used by performing random walk and Levy's flight has to be applied based on the best nest that has been carried for next generation. kth Ant moves are performed from m to n. Apply updation of pheromone trails, once search has been performed. Global updation of pheromone has to be carried out and hence pheromone trail updation is performed. Perform iteration, in this step it accumulates entire iterations until all jobs have been scheduled. It will list all the necessary resources for virtual machines.

A new Hybrid algorithm perform search is much faster than rest of all the optimization algorithm, for job scheduling problems resources need to allocated to job with the limited interval, thereby allocating resources to other jobs is much easier so that no jobs need to wait for resource *n* number of times. By searching the required resources and allocating to jobs a new Hybrid algorithm performance becomes much better which in turn leads to reduction of energy consumption and Makespan.

4.4. Flow chart

Fig. 2 shows the flow chart of Hybrid algorithm, thus initialization of pheromone, heuristic information, number of nest and random initial solution has to be done. The jobs that have to be done by the colony of ants are determined. For processing of next job, transition rule have to be applied. Construction of ant scheduling for each and every ant is carried out, that is which ant has to execute first is scheduled. Finding resources for job scheduling in cloud computing has been performed using the cuckoo search process, since cuckoo search is very easy to implement that is local search in ant colony optimization is performed using cuckoo search. Trail of pheromone is updated using a new solution and global updation is also carried out. Once local search and other non-local are performed process is terminated.

Local search: the current best nest which has been carried out from the past generation is fetched. Condition function is evaluated for checking fitness with maximum generation, if condition is satisfied cuckoo value is fetched and levy's flight is applied. Evaluation of quality/fitness is carried out and a random nest is chosen, if the fitness is greater than random nest that has been chosen. The value of new nest has been replaced, construction of nest is taken placed and ranking is given to them. The best solution from current best nest is carried out to next generation.

5. Experimental analysis

Computational and data-demanding problem can be solved with the help of simulation tools; simulation tool is created comprising parallel execution and distributing computing, using tools jobs which are created for parallel execution as it has been carried out by virtual machines in cloud computing. A cloud computing lab has been setup to analyze the performance of an algorithm. The time taken for the following factors such as job creation time, tasks creation time, result retrieval time and destruction time are determined. Based on these factors the total execution time has been evaluated for analyzing the performance of an algorithm and the analysis is shown in Tables 2–4.

Table 2 shows job creation time, tasks creation time, result retrieval time and destruction time for Task 1, Task 2 and Task 4 which are evaluated for five consecutive iterations and mean value is evaluated. Table 3 shows job creation time, tasks creation time, destruction time and result retrieval time which is evaluated up to 32 tasks.

Job creation time: job creation time is stated as the time taken for creating a new job for execution. Generally job manager includes remote call and time taken to allocate space in the database by a job manager. In some schedulers generally job creation time is states as writing some files to disk.

$$J_{ct} = R_c + D_s \text{ (or) } J_{ct} = D_w \tag{21}$$

Fig. 3 shows the job creation time of a new Hybrid algorithm. As the number of tasks increases like 2, 4, 8, 16 and 32, job creation time also increases slightly. It is clear from the figure that as the number of tasks increases, variation in job creation time is minimal and it is not drastic.

Job submission time: job submission time can be stated as the time taken for submission of job to the job manager, in other words time taken to start the execution of the job in the database. In case of schedulers, it is the time taken for execution of tasks that has been created.

$$J_{st} = S_t \text{ (or) } J_{st} = E_j \tag{22}$$

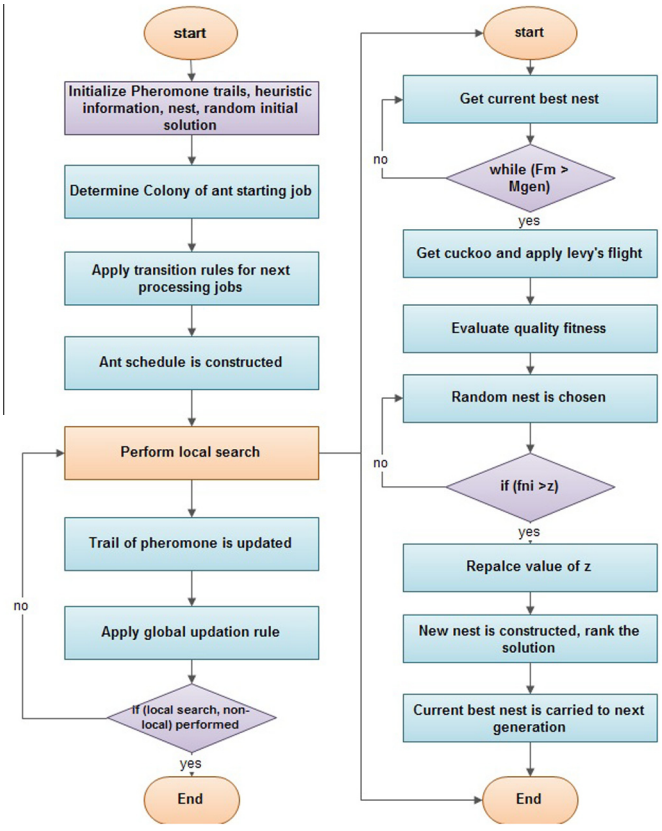

Figure 2 Hybrid algorithm flow chart.

Table 2 Job creation time, task creation time, result retrieval time, job destruction time in terms of iteration.

Sl. No.	Job creation time			Task creation time			Result retrieval time			Job destruction time		
	Task 1	Task 2	Task 4	Task 1	Task 2	Task 4	Task 1	Task 2	Task 4	Task 1	Task 2	Task 4
1	0.0260	0.0290	0.0294	4.2042	4.2083	4.2113	0.0100	0.0104	0.0105	0.0256	0.0269	0.0262
2	0.0285	0.0289	0.0265	4.2021	4.2036	4.2052	0.0102	0.0103	0.0107	0.0266	0.0263	0.0268
3	0.0261	0.0254	0.0258	4.2030	4.2034	4.2036	0.0104	0.0110	0.0105	0.0255	0.0256	0.0261
4	0.0269	0.0272	0.0281	4.2142	4.2125	4.2380	0.0103	0.0104	0.0122	0.0239	0.0256	0.0262
5	0.0265	0.0251	0.0269	4.2112	4.2122	4.2133	0.0101	0.0105	0.0108	0.0255	0.0257	0.0260
Total	0.134	0.1356	0.1367	21.0347	21.0400	21.0714	0.0510	0.0526	0.0547	0.1271	0.1301	0.1313
Average	0.0268	0.02712	0.02734	4.20694	4.2080	4.21428	0.0102	0.01052	0.01094	0.02542	0.02602	0.02626

Task creation time: task creation time is stated as the time taken for creation of tasks and saving disk information. The job manager will try to save required task information in its database. In case of scheduler, task creation time is mentioned as time taken to save task information in a file on the file system.

$$T_{ct} = C_t + t_{\sin f} \tag{23}$$

Fig. 4 shows the task creation time of Hybrid algorithm; as the number of tasks increases from 2, 4, 8, 16 and 32 the time taken for the creation of tasks also increases.

Result retrieval time: result retrieval time is stated as the time taken to retrieve the result from the job manager and display it to the client. In the job manager result retrieval time is mentioned as the time taken for obtaining results from database. In case of other schedulers, result retrieval time is represented as time taken to read from file system.

$$R_{rt} = D_{jrt} \tag{24}$$

Fig. 5 shows the result retrieval time of a new Hybrid algorithm; as the number of tasks increases like 2, 4, 8, 16 and 32 the result retrieval time also increases.

Job destruction time: job destruction time is stated as the time taken for destruction of job or the time taken for deletion of the entire job and its associated information present in the database. Scheduler job destruction time is the time for completely deleting job and task information.

$$J_{dt} = J_{dt} + t_{d\inf} \tag{25}$$

Table 3 Job creation time, task creation time, result retrieval time, job destruction time.

Sl. No.	No. of tasks	Hybrid algorithm			
		Job creation time	Task creation time	Result retrieval time	Job destruction time
1	1	0.0268	0.03312	0.0102	0.02542
2	2	0.02712	0.03318	0.01052	0.02602
3	4	0.02734	0.03458	0.01094	0.02626
4	8	0.02766	0.036038	0.011283	0.02687
5	16	0.02799	0.037558	0.011637	0.027513
6	32	0.02833	0.039142	0.012002	0.028162

Table 4 Comparison of speed-up of Hybrid algorithm and ACO algorithm.

Sl. No.	No. of tasks	Speed-up (s)	
		ACO algorithm	Hybrid algorithm
1	1	4.20697	4.20697
2	2	4.21	4.21
3	4	4.21756	4.21756
4	8	4.2279	4.2279
5	16	4.27572	4.27572
6	32	4.2829	4.2829
7	64	4.29332	4.29332
8	128	4.29342	4.29342
9	256	4.29805	4.29805

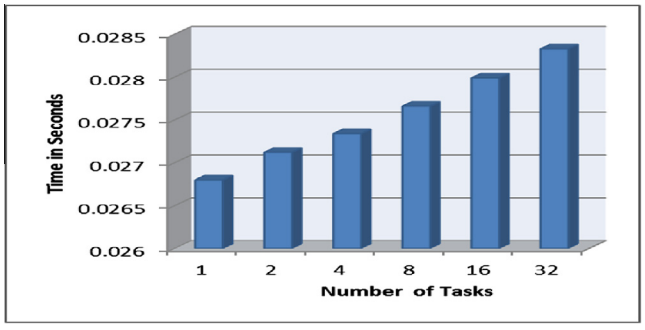

Figure 3 Job creation time based on number of tasks.

Fig. 6 shows the job destruction time of Hybrid algorithm; as the number of tasks increases from 2, 4, 8, 16 and 32 the job destruction time also increases considerably.

Total time: total time is the complete time taken to perform the job creation time, task creation time, job submission, job waiting time, task execution time, result retrieval time, job destruction time.

$$T_t = (J_{ct} + T_{ct} + J_{st} + J_{wt} T_{et} + R_{rt} + J_{dt}) \quad (26)$$

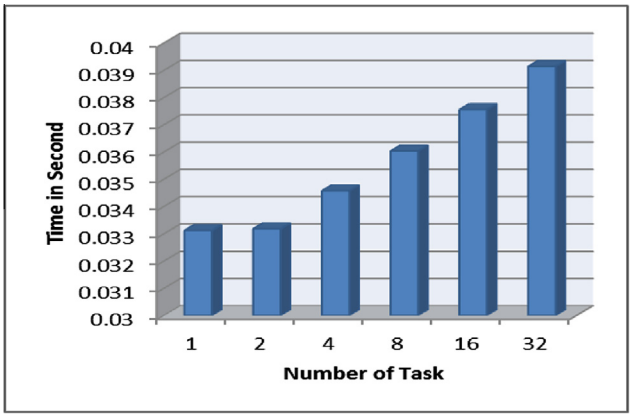

Figure 4 Total time for task creation based on number of tasks.

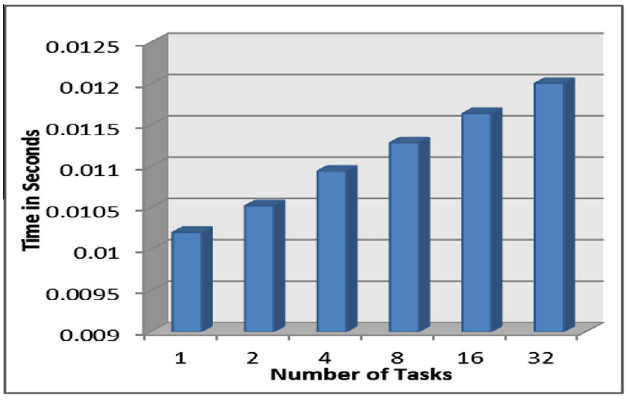

Figure 5 Result retrieval time based on number of tasks.

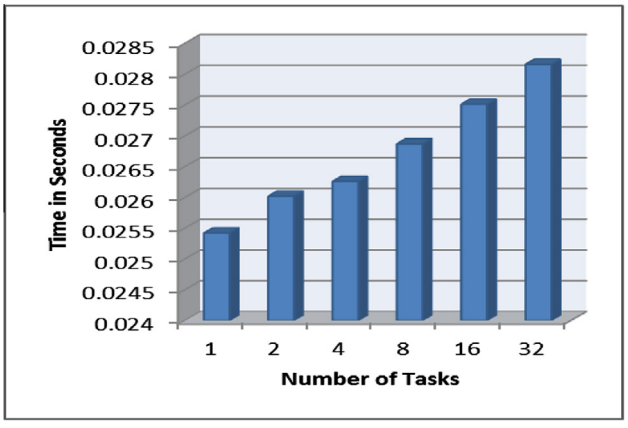

Figure 6 Job destruction time based on number of tasks.

Fig. 7 shows the speed-up comparison of a new Hybrid and ACO algorithm; if the number of tasks increases speed-up time also increases considerably, from Fig. 7 it is clear that the performance of a new Hybrid algorithm tends to be higher than that of ACO algorithm. Fig. 7 depicts that if tasks increase the performance of a new Hybrid and ACO algorithm same up to 16 tasks and when the number of tasks increased to

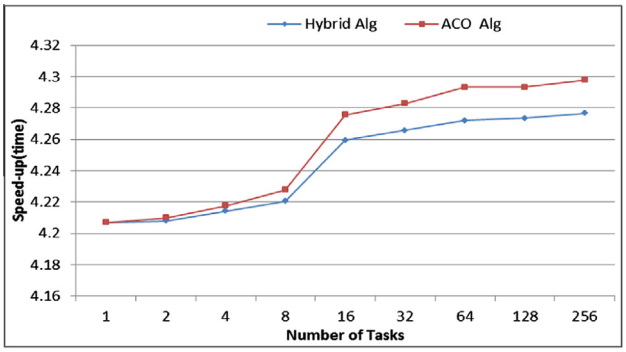

Figure 7 Speed-up of Hybrid algorithm and ACO algorithm.

128 and 256 the performance of Hybrid algorithm is higher than that of ACO algorithm. This is because resource scheduling is performed better than in ACO algorithm.

Further a cloud computing lab has been step up and we have analyzed the execution time, energy consumption, energy improvement rate and make span of Hybrid algorithm with ACO algorithm. The analyses are shown in Tables 5, 6 and from analysis it seems to be clear that energy consumption for Hybrid algorithm is considerably reduced. Table 5 depicts the comparison of algorithms on the basis of factors such as execution time, energy consumed, energy improvement rate and makespan improvement rate.

Speed-up comparison: Table 4 shows performance analysis of a new Hybrid and ACO algorithm. Evaluation is performed up to 256 tasks using the tools.

Fig. 8 shows performance comparison of a new Hybrid and ACO algorithm executed in the cloud computing environment. The performance of a new Hybrid and ACO algorithm is the same up to 35–40 tasks in real time environment, from tasks 45 and the performance of a new Hybrid algorithm is substantially better than that of ACO algorithm. This is mainly due to faster searching of new Hybrid algorithm, at tasks 35–40 both search equally; as the number of tasks increases there is a gradual decrease in ACO compared with Hybrid algorithm.

Energy consumption: Fig. 9 shows the comparison of energy consumption graph, as the number of tasks increases to 40, 50, 60 and 70. The consumption of energy by a new Hybrid algorithm is not as much of ACO algorithm.

Energy utilization: Fig. 10 shows the energy utilization graph of a new Hybrid algorithm and ACO algorithm. The energy consumed by the Hybrid algorithm is considerably less than the ACO algorithm. Moreover as the number of tasks

Table 5 Comparison of Hybrid algorithm and ACO algorithm.

Sl. No.	No. of tasks	ACO algorithm				Hybrid algorithm			
		Speed - up (s)	Energy consumed (J)	Energy improvement (%)	Makespan improvement (%)	Speed - up (s)	Energy consumed (J)	Energy improvement (%)	Makespan improvement (%)
1	5	1.3	170.69	34.13	26	1.3	170.69	34.13	26
2	10	2.4	316	31.6	24	2.4	316	31.6	24
3	15	4.05	533.25	35.55	27	4	521.4	34.76	26
4	20	5.25	691.25	34.56	26.25	5	655.7	32.78	25
5	25	6.38	839.77	33.56	25.25	6	790	31.6	24
6	30	7.53	991.45	33.04	25.31	7	921.14	30.7	23.3
7	35	9.09	1196.85	34.19	25.91	8	1053.07	30.08	22.8
8	40	10.29	1354.8	33.87	25.72	9	1185	29.62	22.5
9	45	11.49	1512.8	33.61	25.53	10	1316.14	29.24	22.2
10	50	13.09	1722.9	34.45	26.1	11	1445.7	28.91	22
11	55	14.29	1880.99	34.19	25.92	12	1580	28.72	21.81
12	60	15.04	1979.74	32.99	25.06	13	1706.4	28.44	21.66
13	65	16.24	2137.74	32.88	24.98	14	1840.7	28.31	21.53
14	70	17.44	2295.74	32.79	24.91	15	1975	28.21	21.43

Table 6 Comparison of energy consumption based on No. of processors.

Sl. No.	No. of Processor	ACO algorithm		Hybrid algorithm	
		Energy consumed (J)	Energy improvement (%)	Energy consumed (J)	Energy improvement (%)
1	1	103.68	99	98.75	98.75
2	2	190	82.64	156	78
3	3	277	75.96	213	71.1
4	4	364	70.12	270.6	67.65
5	5	451	67.82	327	65.58
6	6	538	65.28	385.2	64.2
7	7	625	65.19	442.5	63.2
8	8	712	64.36	499.8	62.4
9	9	799	63.27	557.1	61.9
10	10	886	63.12	614.4	61.44

increases, the energy utilized by Hybrid algorithm decreases considerably and at the particular number of tasks energy consumption by Hybrid algorithm continues to be in the steady state. Even though the Hybrid algorithm searches much faster than ACO algorithm up to tasks 50 it gradually reduces the energy consumption; as the number of tasks increases more and more, further reduction of energy is negligible.

Makespan: Fig. 11 shows makespan improvement comparison graph of Hybrid algorithm and ACO algorithm. The comparison shows that makespan of Hybrid algorithm reduces considerably than that of ACO algorithm on the basis of tasks.

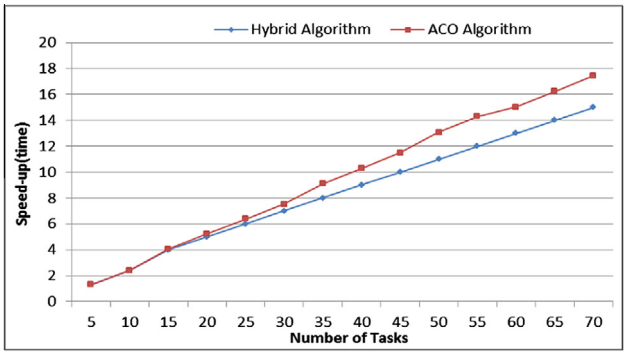

Figure 8 Speed-up of Hybrid algorithm and ACO algorithm.

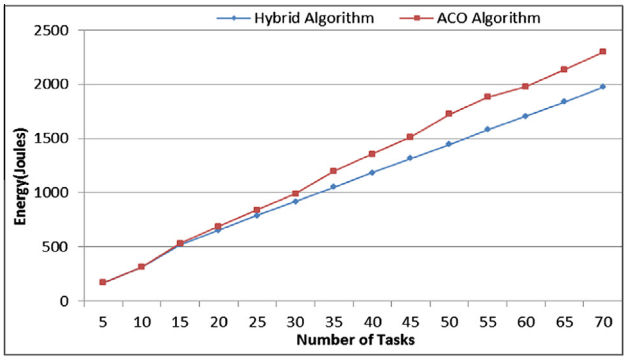

Figure 9 Energy consumption of Hybrid algorithm and ACO algorithm.

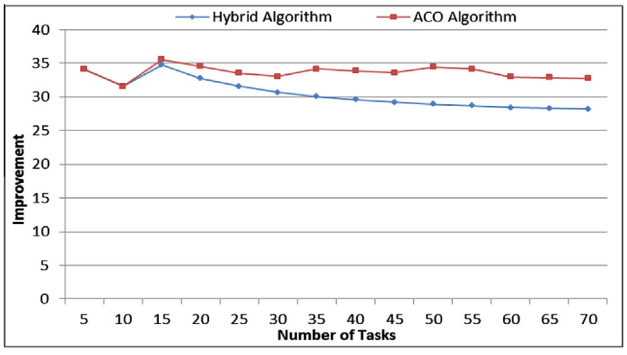

Figure 10 Energy utilization of Hybrid algorithm and ACO algorithm.

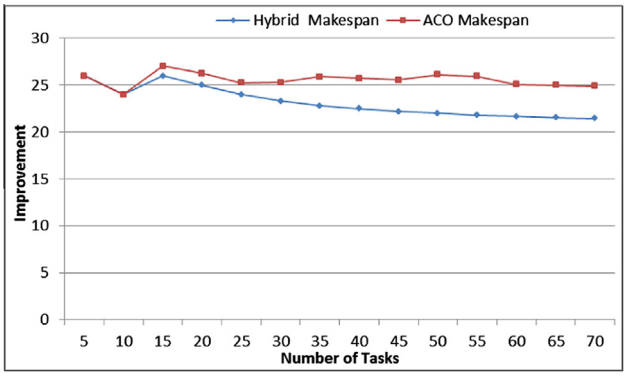

Figure 11 Makespan of Hybrid algorithm and ACO algorithm.

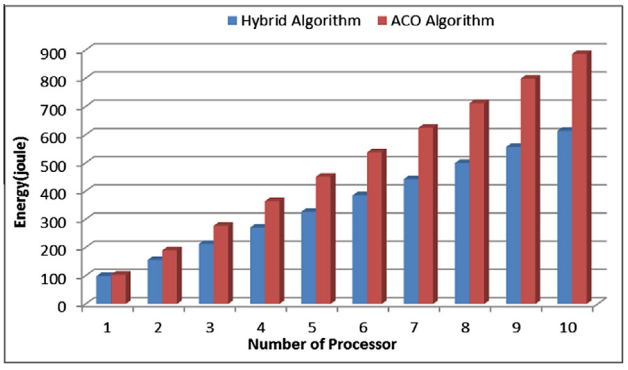

Figure 12 Energy consumption of Hybrid algorithm and ACO algorithm based on number of processors.

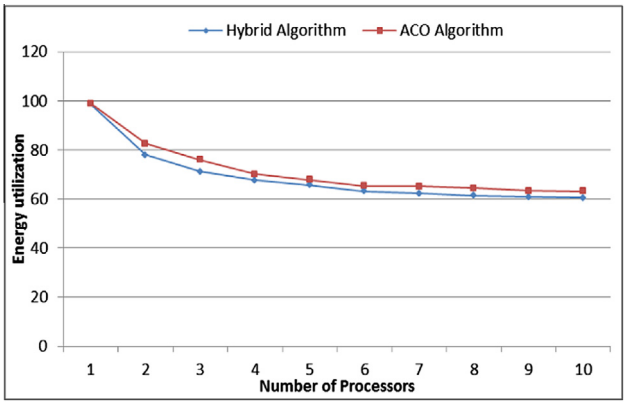

Figure 13 Energy improvement based on No. of processors.

Energy consumption of processor: Table 6 shows comparison of energy consumed by a new Hybrid algorithm and ACO algorithm with respect to processors.

The Fig. 12 shows energy consumption comparison graph of a new Hybrid algorithm, when processors are increased to 6, 7, 8, 9 and 10. The consumption of energy by Hybrid algorithm is low, when compared to ACO algorithm.

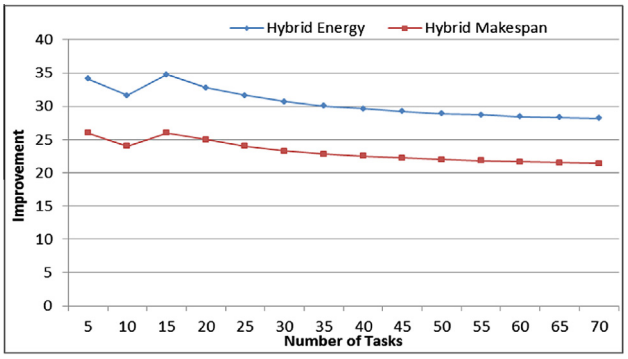

Figure 14 Makespan and energy utilization of Hybrid algorithm.

Energy improvement of processor: Fig. 13 shows energy improvement of a new Hybrid algorithm and ACO algorithm. Energy consumed by Hybrid algorithm is considerably less than ACO algorithm, when processors increase energy utilized by Hybrid algorithm also decreases considerably and at the particular number of tasks energy consumption by Hybrid algorithm continues to be in the steady state.

Energy and makespan improvement of Hybrid algorithm: Fig. 14 shows makespan and energy improvement of a new Hybrid algorithm and it is clear that energy consumed by a Hybrid algorithm significantly decreases in regard to number of tasks compared with ACO algorithm. Moreover, makespan of Hybrid algorithm is also reduced.

6. Conclusion

Cloud computing provides computing as a service afore product as a service. In cloud computing shared software resources and vital information are provided to the computer on basis of usage in a network. The ultimate goal of energy efficient scheduling is to reduce cost and computing infrastructure. In this paper, a new Hybrid algorithm is proposed for reduction of energy consumption and makespan. The execution time of Hybrid algorithm is evaluated in cloud computing lab as the number of tasks increases, the time taken for Hybrid algorithm is less compared to the ACO algorithm. Energy consumed is calculated and the improvement rate is compared with ACO algorithm with respect to number of tasks. It is very clear that energy consumed for job scheduling using Hybrid algorithm decreases considerably. This is possible only if the makespan of job is decreased. Further the energy consumption based on number of processors is analyzed and it depicts that energy consumed is less as the number of processors goes on increasing. Energy consumption seems to be at steady state if we increase the number of tasks and processors. In the future, we plan to extend the work for 1000 to 10,000 jobs to evaluate energy consumption.

References

Ahn, Chang Wook, An, Jinung, Yoo, Jae-Chern, 2010. Estimation of particle swarm distribution algorithms: combining the benefits of PSO and EDAs. Inf. Sci. 192, 109–119.

Bacigalupo, David A., van Hemert, Jano, Chen, Xiaoyu, Usmani, Asif, Chester, Adam P., He, Ligang, Dillenberger, Donna N., Wills, Gary, Gilbert, Lester, Jarvis, Stephen A., 2011. Managing dynamic enterprise and urgent workloads on clouds using layered queuing and historical performance models. Simul. Model. Pract. Theory 19, 1479–1495.

Bae, Changseok, Yeh, Wei-Chang, Chung, Yuk Ying, Liu, Sin-Long, 2010. Feature selection with Intelligent Dynamic Swarm and Rough Set. Expert Syst. Appl. 37 (10), 7026–7032.

Byun, Eun-Kyu, Kee, Yang-Suk, Kim, Jin-Soo, Maeng, Seungryoul, 2011. Cost optimized provisioning of elastic resources for application workflows. Future Gener. Comput. Syst. 27 (8), 1011–1026.

Dhavachelvan, P., Uma, G.V., 2005. Multi-agent based framework for intra-class testing of object-oriented software. Int. J. Appl. Soft Comput., Elsevier 5, 205–222.

Dhavachelvan, P., Uma, G.V., Venkatachalapathy, V.S.K., 2006. A new approach in development of distributed framework for automated software testing using agents. Int. J. Knowl.-Based Syst., Elsevier 19, 235–247.

Rubing Duan, R. Prodan, T. Fahringer, Performance and cost optimization for multiple large-scale grid workflow applications, Proceedings of the 2007 ACM/IEEE Conferences on Supercomputing, 2007, pp. 1–12.

Ferrandi, Fabrizio, Lanzi, P.L., Pilato, C., Sciuto, D., 2010. Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 29 (6), 911–924.

Garg, Saurabh Kumar, Yeo, Chee Shin, Anandasivam, Arun, Buyya, Rajkumar, 2011. Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. J. Parallel Distrib. Comput. 71 (6), 732–749.

Guo, Suchang, Huang, Hong-Zhong, Wang, Zhonglai, Xie, Min, 2011. Grid service reliability modeling and optimal task scheduling considering fault recovery. IEEE Trans. Reliab. 60 (1), 263–274.

Hu, Xia-Min, Zhang, Jun, Chung, H.S., Li, Yun, 2010. SamACO: variable sampling ant colony optimization algorithm for continuous optimization. IEEE Trans. Syst. Man Cybern. B Cybern. 40 (6), 1555–1566.

Huang, Ye, Bessis, Nik, Norrington, Peter, Kuonen, Pierre, Hirsbrunner, Beat, 2013. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm. Future Gener. Comput. Syst. 29 (1), 402–415.

Losup, Alexandru, 2011. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans. Parallel Distrib. Syst. 22 (6), 931–945.

Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y.C., Talbi, E.G., Zomaya, A.Y., Tuyttens, D., 2011. A Parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. 71 (11), 1497–1508.

Montero, Ruben S., Moreno-Vozmediano, Rafael, Lorente, Ignacio M., 2011. An elasticity model for High Throughput Computing clusters. J. Parallel Distrib. Comput. 71 (6), 750–757.

Moreno-Vozmediano, Rafael, Montero, R.S., Llorente, I.M., 2011. Multicloud deployment of computing clusters for loosely coupled MTC applications. IEEE Trans. Parallel Distrib. Syst. 22 (6), 924–930.

Rajeswari, M., Sambasivam, G., Balaji, N., Saleem Basha, M.S., Vengattaraman, T., Dhavachelvan, P., 2014. Appraisal and analysis on various web service composition approaches based on QoS factors. J. King Saud Univ. – Comput. Inf. Sci. 26, 143–152.

Sha, D.Y., Lin, Hsing-Hun, 2010. A Multi-objective PSO for job-shop scheduling problems. Expert Syst. Appl. 37 (2), 1065–1070.

SiYuan, Jing, 2013. A novel energy efficient algorithm for cloud resource management. Int. J. Knowl. Lang. Process. 4 (2), 12–22.

Sumathi, Surekha, 2010. PSO and ACO based approach for solving combinatorial Fuzzy Job Shop Scheduling. Int. J. Compt. Technol. Appl. 2 (1), 112–120.

Qi Tan, Hua-Ping Chen, Bing Du, Xiao-lin Li, Two-agent scheduling on a single batch processing machine with non-identical job sizes, Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011, pp. 7431–7435.

Tao, Qian, Chang, Hui-you, Yi, Yang, Gu, Chun-qin, Li, Wen-jie, 2011. A rotary Chaotic PSO algorithm for trustworthy scheduling of a grid workflow. Comput. Oper. Res. 38 (5), 824–836.

Tavakkoli-Moghaddam, R., Azarkish, M., Sadeghnejad-Barkousaraie, A., 2011. A new hybrid multi-objective Pareto archive PSO algorithm for a bi-objective job shop scheduling problem. Expert Syst. Appl. 38 (9), 10812–10821.

Venkatesan, S., Saleem Basha, M.S., Chellappan, C., Vaish, Anurika, Dhavachelvan, P., 2013. Analysis of accounting models for the detection of duplicate requests in web services. J. King Saud Univ. – Comput. Inf. Sci. 25, 7–24.

Wang, Wei, Zeng, Guosun, Tang, Daizhong, Yao, Jing, 2012. Cloud-DLS: dynamic trusted scheduling for cloud computing. Expert Syst. Appl. 39 (3), 2321–2329.

Zhao, Chunyan, Xu, Baomin, Hu, Enzhao, Hu, Bin, 2011. Job scheduling algorithm based on Berger model in cloud environment. Adv. Eng. Softw. 42 (7), 419–425.

Yang, Xueming, Yuan, Jiangye, Yuan, Jinsha, Mao, Huina, 2010. An improved WM method based on PSO for electric load forecasting. Expert Syst. Appl. 37 (12), 8036–8041.

Yeo, Sungkap, Lee, Hsien-Hsin S., 2011. Using mathematical modeling in "Provisioning a Heterogeneous Cloud Computing Environment". IEEE Comput. Soc. 44 (8), 55–62.

Yuan, Yulai, Wu, Yongwei, Wang, Qiuping, Yang, Guangwen, Zheng, Weimin, 2012. Job failures in High Performance computing systems: a large-scale empirical study. Comput. Math. Appl. 63 (2), 365–377.

Zhiqiang Zhang, Jing Zhang, Shujuan Li, A Modified Ant Colony Algorithm for the Job Shop Scheduling Problem to Minimize Makespan, International Conference on Mechanic Automation and Control Engineering (MACE), 2010, pp. 3627–3630.