Communication-aware and Energy-efficient Resource Provisioning for Real-Time Cloud Services

Aboozar Rajabi, Hamid Reza Faragardi and Nasser Yazdani
School of Electrical and Computer Engineering
University of Tehran
Tehran, Iran
{ab.rajabi, h.faragardi, yazdani}@ut.ac.ir

Abstract—**Operating expense of data centers is a tremendous challenge in cloud systems. Energy consumption of communication equipment and computing resources contributes to significant portion of this cost. In this paper, an online energy-aware resource provisioning framework to minimize the deadline miss rate for real-time cloud services is introduced. Communication-awareness is also taken into consideration in order to reduce energy consumption of the network equipment. A wide range of simulation results, based on real data clearly demonstrates noticeable improvement of energy consumption while at the same time the deadline miss rate is less than 1.5% in average.**

Keywords-real-time cloud services; resource provisioning; energy-aware scheduling; communication-awareness.

I. INTRODUCTION

Nowadays, a large number of cost-effective services are supplied by the cloud providers that leads to increase the popularity of such systems [1]. The cost of services is prominent issue which directly affects end user's motivation to utilize the provided services. Therefore, cloud providers should attempt to supply services within acceptable QoS with the minimum possible cost.

Energy consumption is dominant factor which strongly impacts on the cost of services. Consequently, energy reduction will result in decreasing cost of services thereby decreasing operational cost and environmental impacts. As reported by [3], energy consumption of computing servers and network equipment is about 40% of overall data centers' energy consumption. Therefore, to save energy, both host's energy and communication-awareness should be considered together.

Due to dynamic nature of Cloud Computing Systems (CCS), it is almost impossible to apply offline provisioning methods. In contrast, online provisioning algorithms typically require greater design efforts and are conceptually more complex because they should be able to find the optimal solution in the shortest possible execution time. Indeed, designing an online resource provisioning algorithm is a matter of both practical and theoretical interest.

The problem addressed in this paper is to provide a communication-aware solution to minimize energy consumption in the presence of real-time services. Indeed, the resource allocation should be done such that not only consolidate services on fewer hosts but also try to allocate tasks of same services on the same host. The first consolidation provides potential to turn some physical machines off or putting them into sleep mode. The second one minimizes energy consumption of network equipment due to diminish cost of communication between related tasks. In addition, deadlines, precedence stucture and other system constraints such as hosts' memory and disk space should be satisfied to make an acceptable solution.

In this paper, the problem is modeled in a mathematical way and formulated as an Integer Linear Programming (ILP) problem. As ILP-solvers typically take long execution time to find the solution an effective heuristic algorithm is required. In the paper, a new Energy-Aware Imperialist Competitive Algorithm (EAICA) as an online scheduling approach is introduced. EAICA elaborately intensifies the Imperialist Competitive Algorithm (ICA) [2] by interleaving an effective fast local search.

The main contributions of the paper can be stated as follows: 1) Considering heterogeneous servers, 2) Contemplating network's energy consumption besides host's energy, and 3) Proposing a new online scheduler based on the ICA to minimize energy consumed by the precedence-constrained services.

II. RELATED WORKS

As the mentioned problem is NP-hard, various heuristic and meta-heuristic solutions are applied by [5][6][9] to resolve the problem. However, for large scale problems the most of prevalent heuristic algorithms take long execution time. Therefore, they cannot be efficiently utilized as the online scheduler.

The energy-efficient scheduling in real world systems may have various limitations. Nevertheless, the most of previous studies have considered only a few constraints to simplify the problem. For example, they have assumed homogenous hosts [8][10] or deadline-free situations [9].

The network-aware resource allocation has recently emerged in the context of cloud computing. For example, a network-aware scheduler that considers required bandwidth and migration delays has presented by [11]. Also, Kliazovich et. al [4] have underlined role of communication equipment and proposed a scheduling approach which is composed of energy-efficiency and network-awareness. Their approach optimizes the trade-off between task consolidation and distribution of traffic.

III. UNDERLYING MODELS

A. Notations

Table I. The notations used to model the system

Name	Description	Name	Description
N	number of hosts	$\omega(k)$	returns service number which T_k belongs to
H_i	represents ith host	E_i	number of instructions of task T_i
β_i	processor speed of H_i	DL_{ij}	a direct link between $c_i$and c_j
B	number of services	CL_{ij}	a path between $c_i$and c_j
D_i	deadline of service S_i	X	task to processor assignment matrix
m_i	number of tasks that compose S_i	W_{ij}	communication bandwidth of DL_{kl}
M	total number of tasks	CR_{ij}	amount of transmitted data between T_i and T_j
T_k	kth task ($1 \le k \le M$)	TID_i	task interaction density
MEM_i	memory amount of ith server	ϑ_i	a set of prerequisites tasks for T_i
mem_j	memory needed for task T_j	P_i^{max}	maximum power when H_i is fully utilized
STG_i	storage amount of ith server	P_i^{idle}	power consumption of H_i at idle state
stg_j	amount of storage required for task T_j	k_i	fraction of P_i^{max}consumed by H_i at idle state
$C(X)$	cost of assignment X	$TC(X)$	total cost of assignment
$\Delta(X)$	total energy consumption	$TE(X)$	total energy consumption of assignment X
SP	scheduling period	W	number of switches

B. Data Center Topology

The CCS used in this work includes several hosts which are connected together through a two-tier data center network topology. Each host consists of multiple components such as processor, hard disk and memory modules. The topology is consisted of Access and Core Network. The tier-one, Access Network, contains computing servers arranged in some rackes. Each server rack communicate through its rack switch. The core switches at tier-two, enable rack switches to communicate with each other.

C. Service Model

We suppose that a service can be divided into a set of tasks which can be executed concurrently on different hosts. Tasks of the given service require resources including memory, storage space, computation power and a specific communication bandwidth. The tasks of a service can be represented by a Directed Acyclic Graph (DAG). Each node of the DAG displays a task and the arcs show the prerequisite relation. In addition, there is a label on each arc that indicates amount of data that should be transferred between the tasks. Furthermore, each service, $S = \{ S_1, S_2, \dots, S_B \}$, has a certain deadline which its execution must be completed before. It should be noted that, the tasks do not have explicit deadlines.

Although the well-known b-level has some advantages for sorting the tasks, it is not proper for our problem because the tasks come from different services. Consequently, a new metric, namely *laxity*, is defined by

$$Laxity_{T_i} = D_{s_i} - b\_level_{T_i} \quad (1)$$

Considering the $\mathrm{LPath_i} = \{n_i, n_i+1, \dots, n_{exit}\}$, the longest path from node n_i to exit node, the b-level is computed by:

$$\text{b-level}_{T_i} = \sum_{T_j \in \mathrm{node(LPath_i)}} \bar{E}_J + \sum_{e_j \in \mathrm{edge(LPath_i)}} \overline{CR_{IJ}} \quad (2)$$

Therefore, the scheduling algorithm in each processor orders the assigned tasks to that processor according to their laxities in an ascending manner. The algorithm is called Minimum Laxity First (MLF). Resource allocation should be performed in such a way that MLF scheduler on each processor to be able to schedule all tasks and no task misses its deadline.

D. Problem constraints

1) Memory:

$$\sum_{i=1}^{M} mem_i \, x_{ik} \le MEM_k \qquad 1 \le K \le N \quad (3)$$

2) Storage:

$$\sum_{i=1}^{M} stg_i \, x_{ik} \le STG_k \qquad 1 \le K \le N \quad (4)$$

3) Deadline:

$$\sum_{i=1}^{k} \frac{E_i}{\beta_j} x_{ij} \le D_{\omega(k)} \quad 1 \le j \le N, \ 1 \le k \le M \quad (5)$$

4) Task Precedence: We should first present a formula to compute start time of each task as follows:

$$\Phi_X(T_k) = \sum_{j=1}^{N} x_{kj} \sum_{m=1}^{k-1} \frac{E_m}{\beta_j} x_{mj} \quad (6)$$

where $\Phi_X(T_k)$ is the start time of kth task for assignment X. Finish time of each task is equal to its start time plus its execution time. Accordingly, task precedence constraint can be formulated by Eq. 7.

$$F_X(T_i) \le \Phi_X(T_k) \qquad \forall T_i \in \vartheta_k, 1 \le k \le M \quad (7)$$

where $F_X(T_i)$ is finish time of ith task for assignment X.

5) No task redundancy: This paper considers a model where a task must be allocated to exactly one processor.

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad 1 \le i \le M \quad (8)$$

E. Host Power Modeling

The power consumption of a host could be achieved by a linear model defined in Eq. 9 based on the CPU utilization.

$$P(u) = k * P_{max} + (1 - k) * P_{max} * u \quad (9)$$

where $P(u)$ is the power consumption of the host when its CPU utilization is u, P_{max} is the maximum power of a host. It is cost-effective to turn the host off when its utilization is equal to zero. Off-consumption, (i.e., the consumption of plugged-host when it is off) which is 15%

of the idle consumption is also taken into consideration. Accordingly, Eq. 10 can model power of *s*th host.

$$P_{H_s}(X) = \begin{cases} k_s * P_s^{max} + (1 - k_s) * P_s^{max} * u_{H_s}(X) & u > 0 \\ 0.15 * P_s^{idle} & u = 0 \end{cases} \quad (10)$$

where $u_{H_s}(X)$ is utilization of H_s for assignment X and is computed by

$$u_{H_s}(X) = Min\{1, \frac{\sum_{i=1}^{M} \frac{E_i}{\beta_s} x_{is}}{\Psi_s}\} \quad (11)$$

$$\Psi_s = \max_{1 \le i \le M}(F_X(T_i)x_{is}) \quad (12)$$

F. Network Power Modeling

The energy consumption of networks is dominant by switches that form the basis of interconnection networks. According to [12], power consumed by the switch is calculated by the following equation:

$$P_{Switch} = P_{chasis} + n_{linecard} \,.\, P_{linecard} + \sum_{i=0}^{R} n_{ports} \,.\, P_r \quad (13)$$

where P_{chasis} is the base power consumption of a switch related to its hardware, $P_{linecard}$ is the power consumption of an active line-card and P_r corresponds to the power consumed by an active port running at rate *r*. Accordingly, total energy consumptions by the CCS for assignment X can be achieved by

$$TE(X) = \sum_{s=1}^{A} P_s^{PM}(X)\Psi_s + \sum_{i=1}^{W} P_i^{Switch}(X)\,_i \quad (14)$$

where $_i$ is the active time of *i*th switch.

IV. Solution Approach

A. ILP Formulation

We can formulate the problem as an ILP problem:

Minimize $TE(X)$
Subject to (3),(4),(5),(7) and (8) (15)

To incorporate this problem in our solution framework, it is more convenient to integrate the constraints and objective function into a single cost function called total cost. In this way, the goal is only to minimize total cost function. The violation of memory, storage, deadline and task precedence constraints can be represented by the corresponding penalty functions:

$$P_M = \sum_{k=1}^{A} Max(0, \sum_{\forall j \in \varphi_k} \sum_{i=1}^{M} mem_i \, x_{ij} - MEM_k) \quad (16)$$

$$P_S = \sum_{k=1}^{A} Max(0, \sum_{\forall j \in \varphi_k} \sum_{i=1}^{M} stg_i \, x_{ij} - STG_k) \quad (17)$$

$$P_Q = \sum_{p=1}^{N} \sum_{t=1}^{M} Max(0, \sum_{i=1}^{t} E_{ip} x_{ip} - D_{\omega(t)}) \quad (18)$$

$$P_T = \sum_{k=1}^{M} \sum_{\forall i, T_i \in \vartheta_k, 1 \le k \le M} Max(0, \Phi_X(T_i) - \Phi_X(T_k)) \quad (19)$$

We define total cost of an assignment X as weighted sum of total power consumption and all penalties:

$$TC(X) = TP(X) + \gamma_1 . P_M + \gamma_2 . P_s + \gamma_3 . P_Q + \gamma_4 . P_T \quad (20)$$

Where coefficients γ_1, γ_2, γ_3 and γ_4 are used to show importance of each function. In fact, they should be selected in such a way that solving the above-mentioned problem to be equal to minimizing $TP(X)$ while all of constraints to be met. In other words, they should guide the search towards valid solutions. Accordingly, the main goal is minimizing total cost function.

B. Imperialist Competitive Algorithm

ICA was originally proposed [2] to solve continuous optimization problems. ICA is a socio-politically inspired optimization strategy which is used to find a near optimal solution for the problem modeled in section III under following structures.

1) Initial solution: It can be generated randomly or by using a heuristic method. EAICA uses a Simple Heuristic (SH) algorithm. SH works as follows: We know that if the tasks with higher execution times are assigned to the nodes with lower energy consumption then more energy will be saved. We relax deadline and task precedence constraints and just memory and storage are taken into account. SH sorts the tasks with respect to number of instructions in descending manner and also sorts the nodes with respect to their energy consumption in ascending manner. Then the first task is assigned to the first feasible node. SH continues until all of the tasks have been assigned. Then a fast local search is applied to the solution which its pseudo code is given in Alg. I.

2) Assimilation: It is modeled by choosing random tasks from the vectors of colonies and changing their assigned nodes to their corresponding values in the imperialist vector.

3) Revolution: A local search is applied to the imperialists' vectors.

4) Global War: If the best imperialists did not get any better after specific number of iterations, global war performs. A new population is generated randomly and the empires are formed. Then the worst existing empires are replaced by the best new empires and a new world is created.

5) Stopping condition: Algorithm terminates after predetermined number of global wars. The pseudo code of local search and EAICA are provided in Alg. I and II respectively.

Algorithm I: Local Search

Function Local Search (initial solution S, coefficient δ)
S_l = S;
i = ⌊δM⌋;
Select i random tasks; R_0,R_1 ,…, R_i
for each random task
  V= S[R_j] //the node that task R_j is assigned to it;
  Select the best neighbor of S where task R_j is not assigned to node V as S_n;
if (TC (S_n) < TC(S_l))
  S_l = S_n;
return S_l

C. Online Scheduling Algorithm (EAICA)

The online scheduler is invoked to allocate the services that have waited in the execution queue at the start of every scheduling period. Furthermore, unfinished tasks which are remained from previous periods may be migrated from the current host to another based on the scheduler decision. The migrated tasks can continue to their executions on the new hosts. As a result, the algorithm in each period allocates new incoming services besides probably reallocates the older tasks that their executions have not been still finished. It should be mentioned that the migration overhead is ignored in this research similar to [5][7]. Because, it can be imagined infinitesimal for live migration supported by pre-copying task on the destination host before starting its execution.

New incoming services are examined in terms of meeting deadline before putting in the execution queue. If the execution time of the longest task of the incoming service on the fastest host in addition to the remaining time to the next scheduling invocation is larger than deadline of that service, the service is marked as strict. Furthermore, to expedite the scheduler, a new stopping condition is considered in order that ICA can be used as online scheduler in real-world systems. Algorithm is terminated once its execution time becomes more than 10% of the scheduling period. Therefore, the overhead of scheduling is directly dependent on the scheduling period. On the other hand, limiting execution time of EAICA may lead to decrease solution quality or even it may not be able to satisfy all the constraints for heavy workload. Fortunately, as it is seen in the Section V, constraints violation occurs only in few experiments and is less than 1.5% for all services arrived in sequential periods. In addition, scheduler can be run with other services concurrently.

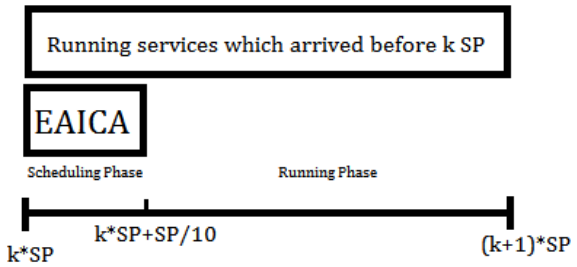

Figure 1. An scheduling period scheme

Fig. 1 illustrates the *k*th scheduling period in the system. Each period is divided into two phases. In the former, ICA is run while the hosts are running the services which arrived in the *(k-2)*th period or earlier. ICA are assigned the services which arrived in the *(k-1)*th period to the hosts. If a strict service arrives in this phase, it will be dropped and is not put in the execution queue. After finishing the first phase, *Running* phase is started in which the scheduler allocates all tasks of the strict services to the feasible hosts as soon as they arrive. Feasible host is a one which has adequate available resources to execute the incoming service besides running other services which have been already allocated to it. Moreover, in both phases if a normal service arrives, it will be added to the execution queue. Consequently, maximum waiting time for normal services is SP + SP/10. Additionally, strict services will certainly meet their deadlines or will be dropped when they arrive in the system.

Algorithm II: EAICA

Randomly initialize the empires Emp_{curr};
Improve empires by local search;
Repeat
  Move the colonies to their relevant imperialists;
  if there is a colony which C_{col}<C_{imp}**then**
    Exchange the positions of that imperialist and colony;
  Revolution among all imperialists;
  Compute the total cost of all empires (TC_{emp});
  Pick the weakest colony from the weakest empire and give to the empire that has the most likelihood to possess it;
  if there is an empire with no colonies **then**
    Eliminate this empire;
  if Global war condition satisfied **then**
    Generate a new empire as New_{emp};
    Sort empires Emp_{curr} and New_{emp} and select the best empires;
until stop condition satisfied

V. Performance Evaluation

Genetic algorithm has been recently applied in the literature to minimize energy consumption [9]. For each problem size, both algorithms (GA and EAICA) have been run 20 times to reach 95% confidence interval. We suppose that arrival rate of services follows Poisson process and its parameter is denoted by service arrival rate per scheduling period in the Tab. II. The presented results belong to 10 continuous scheduling period. We also consider 0.2 as rate of strict service arrival in each period for all simulations. Scheduling period is set to 40 seconds that leads to terminate the algorithm after 4 seconds in order to reach 10% overhead. Energy and deadline miss rate are computed towards each set of hosts and different service arrival rates for both algorithms.

The results of the Tab. II indicate that in terms of solution quality, EAICA significantly outperforms. The energy consumption of EAICA is reduced by 21% in comparison to GA. In addition, results manifest that energy consumption is decreased by 51% in contrast to none power-aware policy where the hosts and switches are not turned-off and consume maximum power. It is noticeable that although energy is increased by incoming new services, the deadline miss rate of EAICA does not grow aggressively. Consequently, it is applicable to satisfy timing constraints specially, for soft real-time services. The deadline miss rate of the proposed EAICA is 16% less than genetic algorithm in average. It is also noticeable that this metric is less than genetic algorithm's deadline miss rate for all the problem sizes.

Table II. Simulation Results

Data Center configuration	Service configuration		EAICA		Genetic Algorithm	
# of hosts	Service arrival rate (per period)	# of tasks	Energy	Deadline miss rate	Energy	Deadline miss rate
60	6	25	0.31164	0.0000	0.51633	0.0000
60	9	30	0.45875	0.0000	0.73328	0.0222
60	10	38	0.59409	0.0204	0.71455	0.0470
80	7	22	0.25202	0.0000	0.47783	0.0130
80	8	36	0.40307	0.0125	0.66171	0.1764
80	11	40	0.61394	0.0272	0.87033	0.2541
100	8	30	0.42475	0.0000	0.64234	0.2837
100	11	44	0.58530	0.0181	0.83320	0.4052
100	12	50	0.77277	0.0333	0.88701	0.3850
Average			0.49073	0.0123	0.70406	0.17628

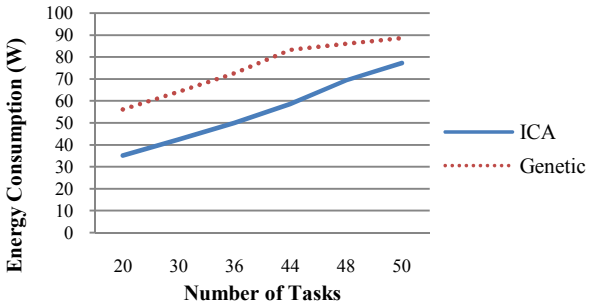

Figure 2. Percentage of Energy Consumption for 100 hosts

The percentage of energy consumption of ICA for 100 hosts and various number of tasks is shown in Fig. 2. The results imply that although number of tasks is increased, the energy consumption increases relatively proportional in comparison with number of hosts.

VI. CONCLUSION

In the paper, a communication-aware model to minimize energy consumption based on proper service allocation is proposed. Indeed, task precedence, service deadlines and other resource constraints were formulated and considered. By integrating energy consumption of servers and network equipment also system constraints, an integer linear programming solution was suggested. Subsequently, a novel online resource scheduler is introduced to find the energy efficient solution while it satisfies real-time requirements as well as other system constraints. EAICA was compared with genetic algorithm for wide range of problem sizes. The simulation results demonstrate that energy consumption of EAICA is reduced by 21% and 51% in comparison to GA and none power-aware policy respectively.

REFERENCES

[1] G. Gruman and E. Knorr, "What Cloud Computing really means," Technical report, Info World Inc., 2008.

[2] E. Atashpaz-Gargari and C. Lucas, "Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition", IEEE Congress on Evolutionary Computation, Singapore, 2007.

[3] Brown, R., et al.: Report to congress on server and data center energy efficiency: public law 109-431. Lawrence Berkeley National Laboratory, Berkeley, 2008.

[4] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," Cluster Computing, vol. 16, no. 1, pp. 65–75, Sep. 2011.

[5] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, 2012.

[6] C. O. Diaz, M. Guzek, J. E. Pecero, and P. Bouvry, "Scalable and energy-efficient scheduling techniques for large-scale systems," in 11th IEEE International Conference on Scalable Computing and Communications (ScalCom), 2011.

[7] R. Buyya, A. Beloglazov, and J. Abawajy. "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," Proceedings of the PDPTA 2010, Las Vegas, USA, July 12-15, 2010.

[8] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," The Journal of Supercomputing, vol. 60, no. 2, pp. 268–280, Mar. 2010.

[9] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, a. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," Journal of Parallel and Distributed Computing, vol. 71, no. 11, pp. 1497–1508, Nov. 2011.

[10] K. H. Kim, R. Buyya, and J. Kim, "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters," Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07), pp. 541–548, May 2007.

[11] Stage, A., Setzer, T., "Network-aware migration control and scheduling of differentiated virtual machine workloads," In Proceedings of the 2009 ICSE Workshop on Software Engineer- ing Challenges of Cloud Computing, International Conference on Software Engineering, IEEE Computer Society,Washington, May 2009.

[12] Mahadevan, P., Sharma, P.,Banerjee, S.,Ranganathan, P., "A power benchmarking framework for network devices," In: Proceedings of the 8th International IFIP-TC 6 Networking Conference, Aachen, Germany, 11-15 May 2009.