IJICTR International Journal of Information & Communication Technology Research ITRC

Volume 5- Number 4- Autumn 2013 (9-18)

From Reliable Distributed System Toward Reliable Cloud by Cat Swarm Optimization

Reza Shojaee
School of Electrical & Computer Engineering
College of Engineering
University of Tehran
Tehran, Iran
r.shojaee@ut.ac.ir

Hamid Reza Faragardi
School of Electrical & Computer Engineering College of Engineering
University of Tehran
Tehran, Iran
h.faragardi@ut.ac.ir

Nasser Yazdani
School of Electrical and Computer Engineering
College of Engineering
University of Tehran
Tehran, Iran
yazdani@ut.ac.ir

Received: January 25, 2013- Accepted: August 25, 2013

Abstract—**Distributed Systems (DS) are usually complex systems composed of various components and cloud is a common type of DSs. Reliability is a major challenge for the design of cloud systems and DSs in general. In this paper an analytical model to analyze reliability in DSs with regards to task allocation was presented. Subsequently, this model was modified and a new model to analyze reliability in cloud systems with regards to Virtual Machine(VM) allocation was suggested. On the other hand, optimal task allocation in DSs is an NP-hard problem, thus finding exact solutions are limited to small-scale problems. This paper presents a new swarm intelligence technique based on Cat Swarm Optimization (CSO) algorithm to find near optimal solution. For evaluating the algorithm, CSO is compared with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The experimental results show that in contrast to PSO and GA, CSO acquires acceptable reliability in reasonable execution time.**

Keywords-distributed system; reliability; cat swarm optimization; cloud computing; task allocation; analytical model.

I. INTRODUCTION

A Distributed System (DS) consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a DS is a parallel application [1]. In distributed computing environment, a parallel application is divided into many tasks, each of which is executed by one or more computers [2]. There are many cases in which the use of a single computer would be possible, but the use of a DS is beneficial for practical reasons. For example, it may be more cost-efficient to obtain the desired level of performance by using a cluster of several low-end computers, in comparison with a single high speed computer. A DS can be more reliable than a non-distributed system, as there is no single point of failure. Moreover, a DS may be easier to expand and manage than a monolithic uniprocessor system [3]. A heterogeneous DS consists of nodes with various computation power and memory capacities. Moreover, connection links which couple the nodes may provide different bandwidths. Such systems provide many advantages over centralized ones, such as improving performance, availability, reliability, resource sharing and extensibility [4]. Tasks of the parallel applications are executed concurrently on different nodes. Distributed System reliability is defined as probability that all the tasks run successfully [5]. Making a distributed system reliable is very important. The failure of a DS can result in anything from easily repairable errors to catastrophic meltdowns. A reliable DS is designed to be as fault tolerant as possible. Fault tolerance deals with making the system function in the presence of faults. Faults can occur in any of the components of a distributed system. Redundancy and diversity are most effective way to achieve reliability [6][7][8][9][10], but they enforce surplus hardware or software costs. Another alternative is optimal task allocation. This approach enhances system reliability just by using optimal task allocation among heterogeneous nodes [11][12][13].

The network topology for our problem is cycle-free such as star, tree and bus. In this paper, we have not considered redundancy, task precedence constraints and transient faults. Solving the optimal task allocation problem for maximizing reliability is known to be NP-hard [11]; therefore, exact methods cannot be used for finding the optimal solution for large scale inputs. This paper presents a meta-heuristic algorithm based on Cat Swarm Optimization to find a near optimal solution within reasonable time. To evaluate the algorithm, PSO, GA and CSO were implemented, and their reliability and execution times were compared for various numbers of tasks and processors. Results indicate that CSO produces more accurate solutions than PSO and GA.

Cloud computing is a large-scale distributed computing paradigm and its applications are accessible at anywhere and anytime. A cloud computing system can be defined as: A scalable distributed computing environment in which a large set of virtualized computing resources, different infrastructures, various development platforms and useful software are delivered as a service to customers as a pay-as-you-go manner usually over the Internet [14]. Actually, it is a type of computing and is usually considered it as a next generation of computing systems. The virtually infinite computing resources on the cloud provider side and the economic reason on the consumer side have made large companies to consider outsourcing their services to cloud. A large number of reputable companies such as Amazon, Google, Verizon, IBM, and Microsoft run and maintain large scale clouds. Cloud computing providers offer their services according to three fundamental types: Infrastructure as a service (IaaS),platform as a service (PaaS), and software as a service (SaaS) where IaaS is the most basic and prevalent type.In this paper after presenting a model for distributed system reliability, we contemplate reliability in IaaS cloud systems and suggest a new model to analyze reliability in IaaS cloud systems. This model can be employed to improve service reliability in such systems.

The remainder of this paper is structured as follows: Related work is summarized in Section II. In Section III we formally defined the problem statement for DS. CSO-based algorithm is presented in section IV. Simulation results to evaluate reliability of DSs are presented in section V. In section VI we introduce an analytical model to analyze reliability in IaaS cloud systems. Section VII reveals the simulation results for cloud computing systems. To show the effectiveness of CSO algorithm for both DS and IaaS cloud we manage the section VIII. Finally, concluding remarks and future work are presented in section IX.

II. RELATED WORK

Shatz et al.[5] defined a model to the problem where the failure of processors or communication links is time-dependent. Based on this scenario, the task with longer execution time will have more failure probability. Many algorithms have been proposed based on this model to find optimal or near optimal solutions. Exact algorithms can produce optimal solutions and are usually based on branch and bound idea. Kartik and Murthy (1995, 1997) used the branch and bound with underestimates and reordered the tasks according to task independence for reducing the computations required. They proved that reliability-oriented task allocation in distributed computing systems is NP-Hard [11]. Thus, exact algorithms only work in problems with small and moderate sizes.

Most studies in recent years have been focused on developing heuristic and meta-heuristic algorithms to solve the problem. In 2001, Vidyarthi and Tripathi proposed a solution based on simple genetic algorithm to find a near optimal allocation quickly [13]. In 2006, Attiya and Hamam developed a simulated annealing algorithm for the problem and compared its performance with branch-and-bound technique [9]. In 2007 Yin et al. proposed a hybrid algorithm combining particle swarm optimization and hill climbing heuristic [12]. In 2010 Kang, et al. used honeybee mating optimization technique [15]. Recently, some prominent studies have been proposed based on modified meta-heuristic algorithms to find optimal or near optimal solutions by proper task allocation[16][17][18]. Also, in another outstanding research, maximizing reliability in real-time distributed systems was stipulated [19].

Although, a lot of studies have been done to analyze reliability in DSs, just a few works have taken cloud computing reliability into account. Chen et al.[20] proposed a security level to achieve trusted cloud. They concentrated to provide reliable migration for virtual machines (VMs).Wu et al. [21] introduced a pipelined approach and a dependence estimation algorithm to improve service reliability in cloud systems. They incorporated an accounting approach in their analysis. Faragardi et al. [22] present an analytical model to evaluate reliability of cloud computing systems. In addition, Vishwanath and Nagappan[23]were examined hardware reliability in Cloud Computing Systems. They investigated server failures and hardware repairs for large data centers and presented a detailed analysis of failure characteristics in such systems. In 2012, Lin and Chang[24]proposed a method to evaluate reliability of the cloud network. They only concentrated on the communication link reliability with considering maintenance budget and time constraints. Some of new researches devoted to the multi-objective optimization in cloud[25]. In our cloud reliability model, we focus on IaaS reliability respect to hazard rate of server and hypervisor.

III. Problem Statement

In a heterogeneous DS, nodes may have different processing speeds, memory sizes, and failure rates. In addition, the communication links may have different bandwidths and failure rates. Another important issue is network topology. The network topology for our problem is cycle-free such as star, tree and bus. Each component of the distributed computing system (node or communication link) can be in any of two states: operational or failed. If a component fails during an idle period, it will be replaced by a spare. We do not consider this to be a critical failure. The failure of a component follows a Poisson process (constant failure rate). Failures of components are statistically independent. This assumption has been widely used in the community of computing system's reliability analysis [26][27][28][29]. Under these assumptions, the reliability of a DS depends on both the number of computing servers composing the system and their individual likelihoods of failure. Obviously the number of tasks is another important factor which affects the results.

Tasks of the given application require certain computer resources such as computational load and memory capacity. They also communicate at a given rate. We are given a set of Mtasks representing a parallel application to be executed on a distributed system with processors [9].

A. *Notation*

The notations used in problem formulation are listed as follows:

- M represents number of tasks.
- N represents number of processors.
- T_i is an *i*th task.
- P_i is an *i*th processor.
- CL_{pq} is a path between node p and q.
- x_{ij} equals one, if and only if T_i is assigned to P_j in the assignment represented by X. Otherwise $x_{ij} = 0$.
- PHR_i is hazard rate for *i*th processor.
- E_{ij} is execution time of T_i on P_j.
- CBW_{ij} is communication bandwidth for CL_{pq}.
- CR_{ij} is communication rate between task i and j.
- PL_{ij} is path load between P_i and P_j.
- CHR_{pq} shows communication hazard rate for CL_{pq}.
- Mem_i is memory amount for P_i.
- mem_i represents essential memory for T_i.
- L_i is processing load for P_i.
- l_i is essential processing load for T_i.
- $R_s(X)$ is system reliability for assignment X.
- $R_{s'}(X)$ is system reliability without considering failure of links.
- $R_{s''}(X)$ is system reliability without considering failure of nodes.
- C(X) is cost of assignment X
- TC(X) is total cost of assignment X

B. *Principle constraints*

The principle constraints for the problem are outlined in this section.

- *Memory:* Memory of each processor is no less than the total amount of memory requirements for all its assigned tasks. This constraint is formulated by Eq. 1.
- *Processing load:* Load of each processor is no less than the total amount of processing load requirements for all its assigned tasks. This constraint is formulated by Eq. 2.
- *Path load:* Load of each path is no less than the total amount of communication rate requirements for all tasks which communicate through this path. This constraint is formulated by Eq. 3.

$$\sum_{i=1}^{M} mem_i x_{ik} \leq Mem_k$$

For all k, $1 \leq k \leq N$ (1)

$$\sum_{i=1}^{M} l_i x_{ik} \leq L_k$$

For all k, $1 \leq k \leq N$ (2)

$$\sum_{i=1}^{M-1} \sum_{j=i+1}^{M} CR_{ij} x_{ip} x_{jq} \leq PL_{pq}$$

For all paths pq, $1 \leq p < q \leq N$ (3)

C. *System modeling*

System is modeled in five parts. In part 1, reliability of nodes is considered. In part 2, reliability of paths is considered. Part 3 formulates system reliability and in part 4, penalty functions are stated. Total cost is suggested in last part.

1. *Reliability of nodes:* We assume reliability of a node is equal to reliability of its processor (i.e., memory and other parts of a node are perfect). Reliability of processor P_k can be achieved from Eq. 4 and due to constant hazard rate [5][30] it reduces to Eq. 5. Then, as the total elapse time for executing the tasks assigned to P_k by assignment X is $\sum_{i=1}^{M} E_{ip} x_{ip}$, the corresponding processor reliability can be computed by Eq. 6. Failure of nodes can be assumed independent [5][30]. Thus reliability of system without considering failure of links can be computed by Eq. 7. Furthermore Eq. 8 is obtained from Eq. 7.
2. *Reliability of paths:* Similarly, the reliability of the path CL_{pq} can be achieved from Eq. 9 and due to constant hazard rate it reduces to Eq. 10. Then, as the total elapse time for communicating the tasks assigned to nodes p and q by assignment X is $\sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}})$, the reliability of path CL_{pq} can be computed by Eq. 11. We assume failures of paths are independent hence reliability of system without considering failure of nodes can be computed by Eq. 12. Furthermore Eq. 13 is obtained from Eq. 12.

$$R_P(t) = e^{-\int_0^t H_P(t)dt} \quad (4)$$

$$R_P(t) = e^{-PHR_p t} \quad (5)$$

$$R_P(X) = e^{-PHR_p \sum_{i=1}^{M} E_{ip} x_{ip}} \quad (6)$$

$$R_{S'}(X) = \prod_{p=1}^{N} e^{-PHR_p \sum_{i=1}^{M} E_{ip} x_{ip}} \quad (7)$$

$$R_{S'}(X) = e^{-\sum_{p=1}^{N} PHR_p \sum_{i=1}^{M} E_{ip} x_{ip}} \quad (8)$$

$$R_{pq}(t) = e^{-\int_0^t H_{pq}(t)dt} \quad (9)$$

$$R_{pq}(t) = e^{-CHR_{pq} t} \quad (10)$$

$$R_{pq}(X) = e^{-CHR_{pq} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}})} \quad (11)$$

$$R_{s''}(X) = \prod_{p=1}^{N-1} \prod_{q=p+1}^{N} e^{-CHR_{pq} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}})} \quad (12)$$

$$R_{s''}(X) = e^{-\sum_{p=1}^{N-1} \sum_{q=p+1}^{N} CHR_{pq} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}})} \quad (13)$$

3. *System reliability:* Due to independence of nodes and path failures, system reliability can be formulated as Eq. 14. Therefore Eq. 15 is obtained from Eq. 14. Maximizing the system reliability is equivalent to minimizing the cost function which is defined in Eq. 16.
4. *Penalty functions:* Penalty function for violating memory, processing load and path load constraints are formulated in Eq. 17, 18 and 19 respectively.
5. *Total cost:* Total cost of assignment X is equal to sum of cost X and all penalties in a weighted manner. Each penalty has a coefficient which shows its importance. Total cost is formulated by Eq. 20.

$$R_s(X) = R_{S'}(X).\ R_{s''}(X) \quad (14)$$

$$R_s(X) = e^{-(\sum_{p=1}^{N} PHR_p \sum_{i=1}^{M} E_{ip} x_{ip} + \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} CHR_{pq} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}}))} \quad (15)$$

$$C(X) = \sum_{k=1}^{N} \sum_{i=1}^{M} PHR_k x_{ik} C_{ik} + \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} CHR_{pq} x_{ip} x_{jq} (\frac{CR_{ij}}{CBW_{pq}}) \quad (16)$$

$$P_M = \sum_{k=1}^{N} Max(0, \sum_{i=1}^{M} mem_i x_{ik} - Mem_k) \quad (17)$$

$$P_L = \sum_{k=1}^{N} Max(0, \sum_{i=1}^{M} l_i x_{ik} - L_k) \quad (18)$$

$$P_C = \sum_{p=1}^{n} \sum_{q=p+1}^{n} Max\{0, (\sum_{i=1}^{M} \sum_{j=1}^{M} x_{ip} x_{jq} CR_{ij}) - PL_{pq}\} \quad (19)$$

$$TC(X) = C(X) + \alpha P_M + \beta P_L + \gamma P_C \quad (20)$$

For determining coefficients two cases can be considered: first, they should scale possible values of P_M, P_L and P_C to comparable ranges to guide the search towards valid solutions and away from invalid ones. Second, decision-maker can tune the value of coefficient with respect to importance of corresponding penalty function. Because of same importance of penalty functions in our model, we assume the coefficients are equivalent. Thus, we replace α,β and γ with a common value, ɳ. Therefore Eq. 20 is rewritten by Eq. 21. The main goal is minimizing Total Cost function which is determined in Eq. 21.

$$TC(X) = C(X) + ɳ\,(P_M + P_L + P_C) \quad (21)$$

IV. CAT SWARM OPTIMIZATION

Cat swarm optimization is a new algorithm developed based on two major behaviors of cats, termed as “seeking” and “tracing”. To apply CSO in the optimization problem, the first step is to decide how many cats to use. Each cat has its own M-dimensional position, velocities for each dimension, a fitness value representing the accommodation of the cat to the fitness function and a seeking/tracing flag to identify whether the cat is in seeking or tracing mode. The final solution would be the best position for one of the cats. CSO keeps the best solution until the end of all iterations [31].

A. Seeking mode

This sub model is used to model the behavior of cat in the period which is resting but looking around and seeking the next position to move. Four essential parameters are defined in seeking mode as Seeking Memory Pool (SMP), Seeking Range of the selected Dimension (SRD), Counts of Dimension to Change (CDC) and Self-Position Consideration (SPC). SMP is defined as the size of seeking memory for each cat, which indicates the points sought by each cat. SRD declares the mutative ratio for the selected dimensions. If a dimension is selected to mutate, the difference between the old value and the new one cannot be out of the range defined by SRD. CDC discloses how many dimensions will be varied. And SPC is a Boolean variable which decides whether the point, on which the cat is already standing, can be one of the candidates to move to. The seeking mode works based on following five steps:

1. Make j copies of the present position of cat_k, where $j = \text{SM}$. If the value of SPC is true, let $j = (\text{SMP} - 1)$, and then retain the present position as one of the candidates.
2. For each copy, according to CDC, randomly plus or minus SRD percent of the present values and replace the old ones.
3. Calculate the fitness values (FS) of all candidate points.
4. If all FS are not exactly equal, calculate the selecting probability of each candidate point by Eq. 22, otherwise set all the selecting probability of each candidate point to 1.
5. Randomly pick the point to move to from the candidate points, and replace the position of cat_k.

$$P_i = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}}, \text{ where } 0 < i < j \tag{22}$$

If the goal of the fitness function is to find the minimum solution, let $FS_{max} = FS_b$, otherwise $FS_b = FS_{min}$.

B. Tracing mode

This sub model is used to model the case which cat is tracing some targets. Once a cat enters the tracing mode, it moves according to its velocities for each dimension. The tracing mode can be described as follows:

1. Update the velocities $V_{k,d}(t)$ for every dimension for the cat_k at the current iteration according to Eq. 23.
2. Check if the velocities are in the range of maximum velocity. If the new velocity is over-range, set it to the limit.
3. Update the position of cat_k according to Eq. 24.

$$v_{k,d}(t) = v_{k,d}(t-1) + r1.c1.\left[x_{best,d}(t-1) - x_{k,d}(t-1)\right] \quad d = 1,2,..,M \tag{23}$$

Where $x_{best,d}(t-1)$ is the position of the cat with the best fitness value at the previous iteration and $x_{k,d}(t-1)$ is the position of cat_k at the previous iteration, c1 is a constant value and r1 is a random value between 0 and 1.

$$x_{k,d}(t) = x_{k,d}(t-1) + v_{k,d}(t) \tag{24}$$

In order to combine two above-mentioned modes into the CSO algorithm, a mixture ratio (MR) of joining the seeking mode and tracing mode must be defined. Clearly, MR is a tiny value, since observations from the behaviors of cats show that they spend most of their waking times on resting and slowly changing their positions. The CSO process can be described in the following seven steps:

1. Create *N* cats in the process.
2. Randomly sprinkle the cats into the *M*-dimensional solution space and randomly select values, which are in the range of the maximum velocity to the velocities of each cat. The position of each cat is improved, using hill climbing algorithm.
3. Then haphazardly pick a number of cats and set them into tracing mode according to MR, and set the others into seeking mode.
4. Evaluate the fitness value of each cat by applying the positions of cats into the fitness function, which represents the criteria of our goal, and keep the best cat into memory.
5. Move the cats according to their flags. If cat_k is in seeking mode, apply the cat to the seeking mode process; otherwise apply it to the tracing mode process.
6. Re-pick number of cats and set them into tracing mode according to MR, then set the other cats into seeking mode.
7. Check the termination condition, if satisfied, terminate the program, and otherwise repeat step 3 through step 6.

V. SIMULATION RESULTS FOR DS

To evaluate the efficiency of the proposed algorithm, intensive experiments have been conducted and the algorithm was compared with PSO and GA. PSO was originally proposed by Kennedy and Eberhart [32], considering the social behavior of natural swarms such as birds, fishes, etc. Similar to communications between swarms in the real world based on the evolutionary computations, PSO combines self-experiences with social experiences. In this algorithm a swarm of particles are randomly generated and each individual improves by referring to experiences of itself and that of the others in the swarm. The swarm intelligence is enriched along with the evolution of each particle and thus the near-optimal solutions can be found. The major components of PSO are Particle Representation, Swarm, Experience and Stopping Criterion. The convergence and parameterization aspects of the PSO are discussed in [33] and [34]. A hybrid of PSO and hill climbing (HPSO) algorithm was applied to solve the problem [12]. To validate the effectiveness of our algorithm, HPSO was implemented and comparison results are listed in Tab. II.

The genetic algorithm was first developed by John H. Holland in the 1960's [35]. In Genetic Algorithm (GA) a population of strings called chromosomes which encode candidate solutions called creatures or individuals to an optimization problem, evolves towards better solutions. The evolution starts from a population of randomly generated individuals and repeats in generations. In each generation, the fitness of each individual is evaluated, several individuals are selected from the current population (based on their fitness) and combined and randomly mutated to form a new population. Then, the new population is used in the next iterations. The algorithm terminates when either a maximum number of generations have been produced or a satisfactory fitness level has been acquired. As the basis for implementation of GA, we assume that GA evolves with a population size of 50 chromosomes and the cross over and mutations rates are 0.8 and 0.1, respectively. These values are determined experimentally from the following ranges. The population size changes from 10 to 100 by the increase of 10. The cross over and mutation rates are both tested in the varying range of 0 to 1 with the increment of 0.1. Also, the maximum number of iterations (stopping condition) is set to 80.

In a DS, many parameters should be determined in order to compute system reliability, such as hazard rate of each node and path, memory and processing load of each node, network topology and etc. System parameters are tabulated in Tab. I. These values are similar to the ones used in [15][12][16]. Moreover, we consider the tree as network topology in the simulations. For each problem size (N, M), 20 simulation runs are conducted by GA, PSO and CSO. The average values of reliability with the corresponding confidence interval at the 95% confidence level and execution time are tabulated in Tab. II. Simulations have shown that CSO results in better reliability in comparison with GA and PSO, while it consumes less execution time. Also, reliability column in Tab. II denotes that our algorithm has smaller reliability deviation rather than GA and PSO

TABLE I SYSTEM PARAMETERS AND THE CORRESPONDING VALUE RANGES

System parameters	Description	Value ranges
E	Task Execution Time	[15, 25]
l	Task Processing Load	[1, 50]
L	Node Processing Load	[100, 200]
mem	Task Memory	[1, 50]
Mem	Node Memory	[100, 200]
CR	Communication Rate	[0, 25]
PL	Path Load	[100, 200]
PHR	Processor Hazard Rate	[0.00005, 0.00010]
CHR	Communication Hazard Rate	[0.00015, 0.00030]
CBW	Communication Bandwidth	[1, 4]

TABLE II EXPERIMENTAL RESULTS FOR VARIOUS NUMBERS OF NODES AND TASKS IN TERMS OF GA, PSO AND CSO

Problem Size		GA		PSO		CSO	
N	M	Reliability	Execution Time (sec)	Reliability	Execution Time (sec)	Reliability	Execution Time (sec)
6	10	0.9893 ± 0.0000	4.250	0.9893 ± 0.0000	4.372	0.9893 ± 0.0000	0.8
6	15	0.9680 ± 0.0002	7.689	0.9792 ± 0.0000	5.839	0.9803 ± 0.0000	2.854
6	20	0.9429 ± 0.0018	16.426	0.9545 ± 0.0007	11.763	0.9730 ± 0.0000	7.285
6	25	0.9345 ± 0.0029	25.662	0.9408 ± 0.0015	19.451	0.9430 ± 0.0002	11.480
8	15	0.9798 ± 0.0018	18.205	0.9818 ± 0.0000	24.362	0.9818 ± 0.0000	7.051
8	20	0.9371 ± 0.0074	52.214	0.9704 ± 0.0015	47.528	0.9741 ± 0.0000	27.878
8	25	0.8982 ± 0.0133	96.859	0.9522 ± 0.0014	72.763	0.9661 ± 0.0007	55.848
8	30	0.8729 ± 0.0374	134.723	0.9311 ± 0.0025	118.763	0.9581 ± 0.0011	82.728

VI. RELIABILITY OF CLOUD

In this section, we strive to modify the reliability model which is proposed in section III in order to analyze reliability in IaaS cloud systems. As we mentioned before, IaaS is a common type of cloud systems in which infrastructures are served as a service. Analyzing reliability in cloud systems is inherently different from distributed system reliability. In the mentioned distributed systems we tackle with task allocation while in IaaS cloud systems the goal is VMs allocation. There are two major differences between the tasks and VMs from the reliability point of view:

1. Each task has a certain execution time on each processor while we do not consider execution time for VMs. Consequently, if no VM is allocated to a server, its reliability is equal to 1 because this server has no effect on system reliability and in this situation we can even turn off the server in order to minimize energy consumption.
2. In spite of distributed systems, In IaaS cloud systems, VMs do not communicate each other. Therefore, the reliability of links could be neglected in the formulation. Although, link hazard rate affect service reliability but for system reliability analysis is not noteworthy.

Based on virtualization concept in IaaS layer, cloud provider installs Virtual Machine Monitor (VMM) on servers. Each VMM could manage number of VMs. By request of customers, one or more VM instances are provisioned with the specific amount of resources. VM, VMM and hardware are three important parts of each server. Although each of which are carefully engineered, they are still capable of failing.We illustrate the virtualization concept of IaaS cloud in Fig. 1.

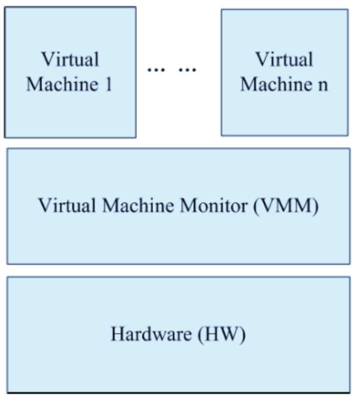

Figure 1. System architecture for the virtualization in IaaS layer of Cloud

Here, we introduce the principle notation of the model:

- VM_i is an ith VM.
- S_i is an ith server.
- M^v represents number of VMs.
- N^S represents number of servers.
- x_{ij}^v equals one, if and only if VM_i is assigned to S_j in the assignment represented by X^v. Otherwise $x_{ij}^v = 0$.
- λ_{HW_i} is hardware hazard rate for ith server.
- λ_{VMM_i} is hypervisor hazard rate for ith server.
- P_i^v is number of dedicated processors for VM_i.
- P_i^s is number of available processors on S_i.
- Mem_i^v is amount of dedicated memory for VM_i.
- Mem_i^s is total amount of memory on S_i.
- ST_i^v is amount of dedicated storage for VM_i.
- ST_i^s is total amount of storage on S_i.
- Ω_i represents the state of S_i and is equal to 1 when this server is on and is equal to 0 when S_i is off.
- $R_{S_i}(X^v)$ is reliability of S_i for assignment X^v.
- $R_{sys}^C(X^v)$ is cloud system reliability for assignment X^v.
- $C_c(X^v)$ is cost of assignment X^v.
- $TC_c(X^v)$ is total cost of assignment X^v .

We assume server failures are independent. Thus, system reliability of the cloud systems can be formulated by Eq. 25.

$$R_{sys}^C(X^v) = \prod_{i=1}^{N^S} R_{S_i}(X^v) (25)$$

To delineate state of each server (on or off) we introduce Ω and it can be determined as follows:

$$\Omega_j = Max(0, x_{ij}^v) 1 \le i \le M^v (26)$$

The reliability of a server can be calculated by Eq. 27 which is similar to Eq. 5. In addition, from Eq. 26,when S_j is off, $\Omega_j = 0$ and thus R_{S_j} =1. As a consequence, with increasing number of off servers, system reliability may enhance. It should be noted that it is not the general case because the servers have not the same reliability.

$$R_{S_j} = e^{-\Omega_j\left(\lambda_{HW\,j} + \lambda_{VMM\,j}\right)t} \quad (27)$$

Furthermore, as we mentioned before the VMs have no certain execution times and so we assign t=1. It leads us to following formulation to calculate server reliability:

$$R_{S_j} = e^{-\Omega_j\left(\lambda_{HW\,j} + \lambda_{VMM\,j}\right)} \quad (28)$$

Finally, from Eq. 25 and 28, system reliability of the cloud system can be formulated by Eq. 29.

$$R_{sys} = e^{-\sum_{j=1}^{N^S}\Omega_j\left(\lambda_{HW\,j} + \lambda_{VMM\,j}\right)} \quad (29)$$

For maximizing R_{sys} we should minimize the following term which called cost function.

$$C_c(X^v) = \sum_{j=1}^{N^S} \Omega_j \left(\lambda_{HW_j} + \lambda_{VMM_j}\right) \quad (30)$$

Penalty functions for violating processing capacity, memory and storage constraints are formulated in Eq. 31, 32 and 33 respectively.

$$P_P = \sum_{j=1}^{N^S} Max\left(0, \sum_{i=1}^{M_v} x_{ij}^v \, P_i^v - P_j^s\right) \quad (31)$$

$$P_{Mem} = \sum_{j=1}^{N^S} Max\left(0, \sum_{i=1}^{M_v} x_{ij}^v \, Mem_i^v - Mem_j^s\right) \quad (32)$$

$$P_{ST} = \sum_{j=1}^{N^S} Max\left(0, \sum_{i=1}^{M_v} x_{ij}^v \, ST_i^v - ST_j^s\right) \quad (33)$$

Total cost of assignment(X^v) is equal to sum of cost (X^v) and all penalties in a weighted manner. Each penalty has a coefficient which shows its importance. Total cost is formulated by Eq. 34.

$$TC_c(X^v) = C_c(X^v) + \alpha_c P_P + \beta_c P_{Mem} + \gamma_c P_{ST} \quad (34)$$

Due to same importance of penalty functions in our model, we suppose the coefficients are equivalent. Thus, we replace α,β and γ with a common value, η. Therefore Eq. 34 is rewritten by Eq. 35. The main goal is minimizing Total Cost function which is determined in Eq. 35.

$$TC_c(X^v) = C_c(X^v) + \eta_c \left(P_P + P_{Mem} + P_{ST}\right) \quad (35)$$

VII. SIMULATION RESULTS FOR IAAS

To assess our CSO-based algorithm for reliability of IaaS cloud, we implementCSO along with two well known meta-heuristic algorithms. First, we concentrate on GA and tune its parameters. 60 chromosomes for population size, uniform crossover and 0.2 for mutation rate are considered as the main parameters for GA which lead to the highest reliability for this algorithm. Moreover, we applied a heuristic algorithm to produce initial solution for GA. It helps to converge the process to the best solution rapidly. For PSO, we set 35 as the number of particles and 0.75 as inertial constant.Furthermore, due to use of Canonical PSOfor this problem,we adjust neighbor component coefficient, cognitive coefficient and social coefficient to zero, 2 and 1.8 respectively. Tab. III shows the computed reliability for these three algorithms. The confidence interval which is shown in the Tab. III is based on 96% confidence level.

Acquired results reveal that CSO is more effective and superior to the Enhanced GA and Canonical PSO in terms of both solution quality and execution time. Fig. 2 indicates the reliability results for 30 servers and various number of VMs for the mentioned above algorithms.

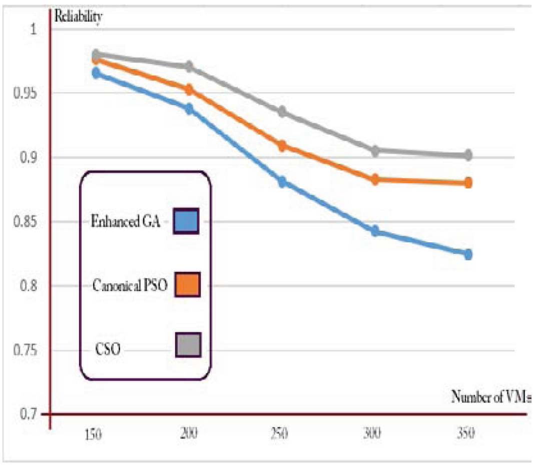

Figure 2. Computed IaaS reliability for 30 servers

Table IV CSO Parameter Settings for DS and IaaS Platforms

Parameter	Description	Values for DS	Values for IaaS
α	Number of cats spread in solution area	30	45
SMP	Number of copies each cat makes in seeking mode	5	7
SRD	Range of variation for each dimension in seeking mode	25%	25%
CDC	Number of dimensions that will change in seeking mode for each copy.	80%	75%
MR	Percentage of cats in tracing mode vs. seeking mode	20%	25%
r1	A random variable used in calculating velocities	[0,1]	[0,1]
c1	A constant variable used in calculating velocities	2	2

VIII. FROM DS TO IAAS BY CSO

In this section, we tackle with the effectiveness of CSO for both problems. The parameter settings for CSO are tabulated into Tab. IV in terms of DS and IaaS platforms. As mentioned before, CSO algorithm has four main parameters in seeking mode including CDC, SRD, SMP and SPC. CDC is the count of tasks/VMs in which processor/server allocation will be changed. SRD is the range of change in processor/server allocation for each task/VM. SMP is the size of memory for each cat which determines number of copies made in seeking process. SPC is always 1 which means current position is considered in experiments. Usually, total number of cats which spread in the solution area depends on the problem size. Few number of cats leads to the less exploration and more exploitation.

Although, the two platforms (DS and IaaS) have quite different components but the conducted simulation indicates the acceptable results which are generated by CSO. Our investigation on the behavior of CSO reveals that the main strong point of CSO in comparison of PSO and GA is its seeking and tracing mode. These modes provide an appropriate trade-off between the exploration and exploitation of the algorithm. Furthermore, seeking mode helps to find better solution in a short period of time.

Table III Experimental results for various numbers of servers and VMs in terms of Enhanced GA, Canonical PSO and CSO

Problem Size		Enhanced GA		Canonical PSO		CSO	
N^S	M^v	Reliability	Execution Time (sec)	Reliability	Execution Time (sec)	Reliability	Execution Time (sec)
20	100	0.9793 ± 0.0001	6.590	0.9831 ± 0.0000	5.712	0.9873 ± 0.0000	2.105
20	150	0.9618 ± 0.0001	10.839	0.9722 ± 0.0000	7.364	0.9804 ± 0.0000	5.933
20	200	0.9493 ± 0.0017	18.216	0.9542 ± 0.0014	10.981	0.9654 ± 0.0001	7.648
30	150	0.9656 ± 0.0024	24.892	0.9763 ± 0.0014	15.753	0.9799 ± 0.0001	13.746
30	200	0.9381 ± 0.0029	32.415	0.9530 ± 0.0023	19.094	0.9708 ± 0.0009	17.297
30	250	0.8814 ± 0.0065	61.198	0.9093 ± 0.0032	32.637	0.9354 ± 0.0010	29.384
30	300	0.8423 ± 0.0106	104.649	0.8827 ± 0.0058	46.260	0.9053 ± 0.0013	41.284
30	350	0.8246 ± 0.0259	194.855	0.8801 ± 0.0062	88.033	0.9014 ± 0.0019	80.472

IX. CONCLUSION

In this paper, we tackle with reliability maximization problem in distributed systems and cloud environment. We first investigate reliability in distributed systems and suggest a mathematical model to analyze reliability. The model consists of two cost functions which manifests the unreliability caused by execution of tasks on the nodes and the other reveals unreliability caused by inter-process communication time. Different penalty functions are also defined to satisfy the application and system constraints. Based on the model we introduce a task allocation algorithm which is inspired from cat behavior. The computational evaluations manifestly support the high performance of our proposed algorithm against other meta-heuristic algorithms which were applied for finding optimal task allocation in DSs. Subsequently, to analyze reliability in IaaS cloud systems, we propose another mathematical model which elaborately represents effect of VMs allocation on cloud system reliability, and apply the CSO to this problem too. The acquired results of conducted simulation indicate that CSO has low deviation from average reliability in all of the cases in contrast to Enhanced GA and Canonical PSO.As a future work we plan to extend our model in order to take PaaS and SaaS into account.

REFERENCES

[1] G. R. Andrews, Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley, 2000.

[2] S. Dolev, Self-Stabilization. The MIT Press, 2000.

[3] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 3rd ed. Addison-Wesley, 2000.

[4] K. K. Aggarwal and S. Rai, “Reliability Evaluation in Computer-Communication Networks,” IEEE Transactions on Reliability, vol. R-30, no. 1, pp. 32–35, Apr. 1981.

[5] S. M. Shatz, S. Member, J. ping Wang, and M. Goto, “Task Allocation for Maximizing Reliability of Distributed Computer Systems,” vol. 11, no. 0, 1992.

[6] A. Elegbede, K. Adjallah, and F. Yalaoui, “Reliability allocation through cost minimization,” IEEE Transactions on Reliability, vol. 52, no. 1, pp. 106–111, Mar. 2003.

[7] C. C. Chiu, Y. S. Yeh, and J. S. Chou, “A fast algorithm for reliability-oriented task assignment in a distributed system,” Computer Communications, vol. 25, no. 17, pp. 1622–1630, Nov. 2002.

[8] C. C. Hsieh, “Optimal task allocation and hardware redundancy policies in distributed computing systems,” European Journal of Operational Research, vol. 147, no. 2, pp. 430–447, Jun. 2003.

[9] G. Attiya and Y. Hamam, “Task allocation for maximizing reliability of distributed systems: A simulated annealing approach,” Journal of Parallel and Distributed Computing, vol. 66, no. 10, pp. 1259–1266, Oct. 2006.

[10] P. A. Tom and C. S. R. Murthy, “Algorithms for reliability-oriented module allocation in distributed computing systems,” Journal of Systems and Software, vol. 40, no. 2, pp. 125–138, Feb. 1998.

[11] S. Kartik and C. Siva Ram Murthy, “Improved task-allocation algorithms to maximize reliability of redundant distributed computing systems,” Reliability, IEEE Transactions on, vol. 44, no. 4, pp. 575–586, 1995.

[12] P. Y. Yin, S. S. Yu, P. P. Wang, and Y. T. Wang, “Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization,” Journal of Systems and Software, vol. 80, no. 5, pp. 724–735, 2007.

[13] D. P. Vidyarthi and A. K. Tripathi, “Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm,” Journal of Systems Architecture, vol. 47, no. 6, pp. 549–554, 2001.

[14] M. Armbrust, A. D. Joseph, R. H. Katz, and D. A. Patterson, “Above the Clouds A Berkeley View of Cloud Computing,” Science, 2009.

[15] Q. M. Kang, H. He, H. M. Song, and R. Deng, “Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization,” Journal of Systems and Software, vol. 83, no. 11, pp. 2165–2174, 2010.

[16] H. R. Faragardi, R. Shojaee, and N. Yazdani, “Reliability-Aware Task Allocation in Distributed Computing Systems using Hybrid Simulated Annealing and Tabu Search,” in 14th IEEE International Conference on High Performance Computing and Communications, 2012, pp. 1088–1095.

[17] R. Shojaee, H. R. Faragardi, S. Alaee, and N. Yazdani, “A New Cat Swarm Optimization based Algorithm for Reliability-Oriented Task Allocation in Distributed Systems,” in Sixth International Symposium on Telecommunications (IST), 2012, pp. 861–866.

[18] H. R. Faragardi, R. Shojaee, M. Mirzazad-Barijough, and R. Nosrati, “Allocation of hard real-time periodic tasks for reliability maximization in distributed systems,” in Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on, 2012, pp. 42–49.

[19] H. R. Faragardi, R. Shojaee, M. A. Keshtkar, and H. Tabani, "Optimal Task Allocation for Maximizing Reliability in Distributed Real-time Systems," in Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on, 2013.

[20] Y. Chen, Q. Shen, P. Sun, Y. Li, Z. Chen, and S. Qing, "Reliable Migration Module in Trusted Cloud Based on Security Level - Design and Implementation," 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, pp. 2230–2236, May 2012.

[21] Z. Wu, N. Chu, and P. Su, "Improving Cloud Service Reliability -- A System Accounting Approach," 2012 IEEE Ninth International Conference on Services Computing, pp. 90–97, Jun. 2012.

[22] H. R. Faragardi, R. Shojaee, H. Tabani, and A. Rajabi, "An Analytical Model to Evaluate Reliability of Cloud Computing Systems in the Presence of QoS Requirements," in Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on, 2013.

[23] K. V. Vishwanath and N. Nagappan, "Characterizing Cloud Computing Hardware Reliability," 2010.

[24] Y.-K. L. and P.-C. Chang, "MAINTENANCE RELIABILITY OF A COMPUTER NETWORK WITH NODES FAILURE IN THE CLOUD COMPUTING ENVIRONMENT," International Journal of Innovative Computing, Information and Control I, vol. 8, no. 6, pp. 4045–4058, 2012.

[25] H. R. Faragardi, A. Rajabi, and R. Shojaee, "Towards Energy-Aware Resource Scheduling to Maximize Reliability in Cloud Computing Systems," in HPCC 2013, 2013.

[26] C. H. Sauer and K. M. Chandy, Computer systems performance modeling, vol. 21. Prentice-Hall Englewood Cliffs, NJ, 1981.

[27] J. F. Lawless, "Statistical models and methods for lifetime data," 1982.

[28] C. Singh, "Calculating the time-specific frequency of system failure," Reliability, IEEE Transactions on, vol. 28, no. 2, pp. 124–126, 1979.

[29] C. S. Raghavendra and S. V Makam, "Reliability modeling and analysis of computer networks," Reliability, IEEE Transactions on, vol. 35, no. 2, pp. 156–160, 1986.

[30] S. Kartik and C. Siva Ram Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems," Computers, IEEE Transactions on, vol. 46, no. 6, pp. 719–724, 1997.

[31] S. chuan Chu, P. wei Tsai, and J. shyang Pan, "LNAI 4099 - Cat Swarm Optimization," pp. 854–858, 2006.

[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Neural Networks, 1995. Proceedings., IEEE International Conference on, 1995, vol. 4, pp. 1942–1948.

[33] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," Evolutionary Computation, IEEE Transactions on, vol. 6, no. 1, pp. 58–73, 2002.

[34] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," Information processing letters, vol. 85, no. 6, pp. 317–325, 2003.

[35] J. H. Holland, "Genetic algorithms," Scientific american, vol. 267, no. 1, pp. 66–72, 1992.

Reza Shojaee received his B.Sc. and M.Sc. degrees both in Computer Engineering from Iran University of Science and Technology and University of Tehran, respectively. Now, he is a researcher of the Router Laboratory at University of Tehran being led by Prof. Nasser Yazdani. His main research interests include reliability and availability analysis, cloud computing and analytical modeling as well.

Hamid Reza Faragardi received his M.Sc. degrees from University of Tehran in 2012 in Computer Engineering. He currently is a PhD student in the real-time department of Malardalen University. His main research interests comprise of dependability modeling, cloud computing and real-time systems. His Ph.D. project is development of AUTOSAR multicore systems and is funded by Volvo and ABB. He could already publish more than eight papers and articles which most of them cover reliability and energy management together in cloud computing systems.

Nasser Yazdani got his B.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran. He worked in Iran Telecommunication Research Center as a researcher and developer for a few years. To pursue his education, he entered Case Western Reserve University, USA, and graduated with a Ph.D. degree in computer science and engineering. Then, he has been working in different companies and research institutes in USA. He joined the School of Electrical & Computer Engineering of University of Tehran, in September 2000 and at present he is a full professor. His research interests include networking, packet switching, access methods, operating systems and database as well.