Neural Comput & Applic
DOI 10.1007/s00521-016-2448-8

ORIGINAL ARTICLE

Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm

Shafi'i Muhammad Abdulhamid[1,3] · Muhammad Shafie Abd Latiff[1] · Syed Hamid Hussain Madni[1] · Mohammed Abdullahi[1,2]

Received: 17 September 2015 / Accepted: 1 July 2016

Abstract In cloud computing, resources are dynamically provisioned and delivered to users in a transparent manner automatically on-demand. Task execution failure is no longer accidental but a common characteristic of cloud computing environment. In recent times, a number of intelligent scheduling techniques have been used to address task scheduling issues in cloud without much attention to fault tolerance. In this research article, we proposed a dynamic clustering league championship algorithm (DCLCA) scheduling technique for fault tolerance awareness to address cloud task execution which would reflect on the current available resources and reduce the untimely failure of autonomous tasks. Experimental results show that our proposed technique produces remarkable fault reduction in task failure as measured in terms of failure rate. It also shows that the DCLCA outperformed the MTCT, MAXMIN, ant colony optimization and genetic algorithm-based NSGA-II by producing lower makespan with improvement of 57.8, 53.6, 24.3 and 13.4 % in the first scenario and 60.0, 38.9, 31.5 and 31.2 % in the second scenario, respectively. Considering the experimental results, DCLCA provides better quality fault tolerance aware scheduling that will help to improve the overall performance of the cloud environment.

Keywords Dynamic clustering · Cloud scheduling · Fault tolerance · Task scheduling · League championship algorithm

✉ Shafi'i Muhammad Abdulhamid
shafii.abdulhamid@futminna.edu.ng

Muhammad Shafie Abd Latiff
shafie@utm.my

Syed Hamid Hussain Madni
madni4all@yahoo.com

Mohammed Abdullahi
abdullahilwafu@abu.edu.ng

[1] Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia

[2] Department of Mathematics, Ahmadu Bello University, Zaria, Kaduna State, Nigeria

[3] Department of Cyber Security Science, Federal University of Technology Minna, Niger State, Nigeria

1 Introduction

Failures are to be expected in cloud computing environment. Cloud resources are known to experience fluxes in their performance delivery [1]. Thus, fault-tolerant scheduling technique that takes care of performance variations, resource fluctuations and failures in the environment is important [2]. As task applications increase to utilize cloud resources for a long time, the task will unavoidably come upon growing amount of component failures [3]. Once task failures occur, it has an effect on the execution of the tasks scheduled to the failed components. Hence, a fault-tolerant mechanism is essential in clouds. Fault tolerance is the capability of the cloud scheduler to safeguard and protect the delivery of projected tasks even with the occurrence of failures in the clouds system [4]. Providing fault tolerance in a cloud computing system, while optimizing resource scheduling and task execution is a demanding task especially for cloud developers. Cloud scheduling mechanisms are expected to have fault-tolerant components that identify failures and resolve them within the shortest possible time. These supportive components allow tasks to be

Published online: 16 July 2016

scheduled on the cloud resources even in case of component breakdown without stopping the applications. In this NP-hard problem, exact solution techniques are at least exponential and a cloud provider has to depend on fairly estimated results obtained in a suitable period based on intelligent algorithms [5].

Fault tolerance awareness has been identified as one of the main issues to ensure reliability, robustness and availability of important services as well as running of applications in the cloud computing system. Task failure may be as a result of many factors such as overloaded RAM, bandwidth shortage or power failure to mention but a few. Fault tolerance depends on either time or hardware redundancy so as to mask task or component failures. Time redundancy involves the re-execution of failed task after the malfunction has been identified. It can be optimized through the application of checkpoint techniques but even with that it still enforces a considerable delay. In various task critical systems, hardware redundancy has been frequently deployed in the outline of task replication to provide fault tolerance, evading delay and sustaining fixed targets. Both methods present downside and negative aspects, re-execution needs extra time and replication needs extra resources, particularly energy. This compels a trade-off between time and hardware redundancy, in IaaS cloud computing systems replication is mostly preferred because response time is normally very vital [6].

The league championship algorithm (LCA) is an intelligent algorithm that is first proposed by Kashan [7]. To form a synthetic championship setting, some idealized rules need to be followed and then introduce the promising computational intelligence algorithm that is modeled based on a number of fascinating results relative to sports championship round robin timetable. To evaluate the performance of the LCA, Kashan and Karimi [8] use five different benchmark test functions that include: the Sphere, Rastrigin, Rosenbrock, Ackley and Schwefel functions. In comparison with other state-of-the-art intelligent algorithms like the particle swarm optimization, simulation results show that the LCA is a reliable optimization algorithm that can converge speedily to the global optimal. This feature and its reliability in avoiding local trappings make it a prepared choice for fault tolerance scheduling in the cloud environment. Consequently, Abdulhamid et al. [9] present LCA-based scheduling technique for global optimization of tasks scheduling in the cloud system. Also, Abdulhamid et al. [10] present a survey of the LCA optimization method detailing the real-world areas that the algorithms have been effectively applied and also explored the prospects and challenges of the technique.

Algorithmic complexity is concerned about how fast a certain algorithm performs taking into cognizance the total operations, input parameters, the resources and time [11, 12]. The LCA has relatively low complexity in terms of the number of iterations and total operations to be performed as compared to other state-of-the-art intelligence techniques. This makes it very easy to implement. The introduction of the DCLCA technique became important as an alternative to the existing approaches because of its strong local search and faster convergence rate. It is versatile in reducing trappings into local optima when solving complex multimodal problems. Additionally, it has few input parameter settings which reduce the uncertainty in choosing the correct values that will lead to fast convergence and accurate results. Furthermore, many of the existing intelligent approaches did not consider fault tolerance strategies and parameters in designing their task scheduling algorithms in the cloud computing system.

The aim of this work is to designing a dynamic clustering league championship algorithm (DCLCA) for task scheduling problem with fault tolerance awareness in cloud computing services. The remaining parts of the article are organized as follows. Section 2 presents the related works, which is divided into dynamic intelligent algorithms used in the cloud environment and fault tolerance aware intelligent algorithms in cloud. Section 3 describes the fitness function of the DCLCA. Section 4 details the fault tolerance scheduling components, task migration and the performance metric. Section 5 shows the design of the proposed DCLCA optimization technique. Section 6 chronicles the experimental setup, while Sect. 7 explains the results and discussion from two different scenarios. Then lastly, Sect. 8 presents the summary and conclusions of our findings.

2 Related works

Fault tolerance awareness in cloud has to do with the mechanisms required to allow a technique to endure system of task execution faults lingering in the system. One of the merits of developing fault tolerance techniques in cloud computing is failure avoidance, healing, cost efficiency and superior performance metrics. As soon as various instances of cloud tasks begin to execute on numerous VMs, and then, some of the servers fail that means there is a fault and it is normally taken care of using fault tolerance mechanism [13]. A number of factors may lead to the failure of a server instance and consequently tasks failure. Many at times, one failure events stimulate another. These factors may include:

hardware malfunction, operating system crashes, network partitions, power outage and unforeseen software performance can all lead to the failure of a server request. Some fault tolerance mechanisms have been presently in harmony with cloud computing scheduling [14, 15]. These include: retry, healing, resubmission, replication, rejuvenation of software, masking and migration to mention but a few, however, most of them are prone to heavy overhead and sometime leads to local trappings. In this section, we review some related literatures that applied intelligent optimization techniques for solving the dynamic task scheduling problem in the cloud computing environment. We also surveyed related works that try to address the issues of fault tolerance using the intelligent algorithms in this environment.

2.1 Dynamic intelligent algorithms in cloud scheduling

Dynamic scheduling techniques are designed to produce a group of independent tasks into a set of suitable computational size or task clusters/partitions using tasks characteristics focusing on priority or a partition [16–18]. It is used to organize the scheduling of clustered tasks over limited heterogeneous cloud resources. A heuristic is applied to categorize the task clusters, with knowledge of the current limited resource states in the cloud environment [16, 19–22]. It also efficiently combines task clustering and mapping into a joint resource allocation technique to improve computing accessibility of the resources.

Gangeshwari et al. [23] introduce a dynamic hyper cubic peer-to-peer cloud (HPCLOUD) which is a structured peer-to-peer framework implemented on a cloud computing system on which MapReduce method is used for tasks scheduling. In addition, fault tolerance can be attained on the HPCLOUD architecture. Experimental outcome shows that the proposed HPCLOUD method has demonstrated superior feat in terms of response time with respect to increase the number of files. However, the dynamicity of HPCLOUD is only at initial level of scheduling and not integrated to cover all levels. In addition, the response time is considered in measuring the fault tolerance parameter, while other important parameters are not taken into account. An intelligent technique based on GA is also presented to decipher the global tasks scheduling problem [24]. The technique is based on the Pareto dominance relationship called NSGA-II, giving no distinct optimal result, but a set of results that are not subjugated on one another. The related outcomes show the efficiency of the presented technique and GA for small- and medium-sized scheduling problems. However, the experimental results did not demonstrate the performance with large and massive cloud-based scheduling problems, the heterogeneity, dynamicity and likely local trappings are not considered.

Using active replication technique, a dynamic and reliability-driven real-time fault-tolerant scheduling algorithm (DRFACS) and greedy algorithm in case of enormous resource failures is put forward by Ling et al. [25]. It aims at increasing reliability by dynamically assign reliant, non-preemptive, non-periodic instantaneous tasks, trying to advance the QoS throughout scheduling process. The experimental result shows that when the computing and communication rate (CCR) value is microscopic, DRFACS scheduling capability is better than FTSA and FTBAR. When value of CCR is increased, DRFACS performance is steadily minimized and FTSA and FTBAR schemes do better than DRFACS trend. This shows that the CCR has a strong impact on DRFACS, FTSA and FTBAR reliability. As the CCR amplifies, DRFACS also steadily progresses the reliability, and it shows that increase in CCR will also increase the reliability of the system. However, the dynamicity of DRFACS technique is prioritized to only critical task. Tawfeek et al. [26] put forward a dynamic cloud job scheduling technique using ant colony optimization (ACO) intelligent optimization method. Random optimization search method is adopted for this approach that is utilized for managing the incoming jobs mapping to the resources. The experimental results indicate that the proposed ACO method outperformed FCFS and RR algorithms. However, the dynamic scheduling in this technique is not a priority in achieving the goal of fault tolerance. Therefore, because of the constant changes in the state of the heterogeneous cloud resources, there is an urgent need for a dynamic clustering scheduling technique to reflect these changes. The dynamic clustering technique will also helps in managing time redundancy, which may also lead to tasks failure at runtime.

2.2 Some recent tasks migration strategies in cloud

A task replication technique is put forward called heterogeneous earliest finish time (HEFT) for task fault tolerance execution in cloud. The proposed HEFT heuristic makes thorough use of further resources for the duration of task replication; it attains good fault tolerance compares to the common replication. HEFT finishes with the small workflows within the short makespan [27]. However, the dynamicity of available cloud resources is not considered when making scheduling decisions.

Bala and Chana [28] put forward hybrid heuristic scheduling approach (HHSA) to schedule scientific workflows tasks on the IaaS cloud. Then, a fault-tolerant method is developed based on VM migration method that transfers the VM routinely in case of job failure occurrences as a result of the overutilization of resources. The simulation outcome indicates that the HHSA performs better than Min–Min, Max–Min, MCT, PSO and GA by tumbling the average makespan for massive scientific workflows such as Cybershake and epigenome. The simulation results demonstrate the efficiency of the proposed method to advance the feat of scientific workflows by significantly tumbling total mean execution time, standard deviation time and makespan. However, the simulation results did not use any fault tolerance parameter for the analysis to arrive at such conclusions. Existing rescheduling techniques for fault tolerance in MapReduce failed to totally reflect on the position of distributed data and the calculation and storage overhead of rescheduling failure jobs [29]. Consequently, a single VM failure will amplify the completion time considerably.

A performance, power and failure–aware relaxed time task execution (PPF-RTTE) algorithm is presented as a performance imposing system, made up of a slowdown estimator and a scheduling technique [30]. The slowdown estimator finds out based on noisy slowdown data models acquired from modern slowdown meters, if jobs will execute within the time limits, invoking the scheduling technique if desired. Experimental outcome show that the proposed approach can be resourcefully incorporated with modern slowdown meters to accomplish tight SLAs in real-world environments, while tumbling set expenditure in just 21 %. However, the PPF-RTTE algorithm is more of a trade-off between fault tolerance and scheduling performance. Also, there is no clear fault tolerance parameter considered in the analysis.

2.3 Fault tolerance aware intelligent algorithms in cloud

Fault tolerance intelligent algorithm-based task scheduling techniques in cloud computing environment are important in order to avoid tasks failure due to heterogeneity and dynamicity of available cloud resources [31, 32]. Min–min based time and cost trade-off (MTCT) is presented for multi-objective workflow scheduling to aid fault recovery in cloud [33]. The MTCT technique was evaluated using simulations with four different real-world scientific workflows scenarios to test the strength of the technique. The outcomes indicate that fault recovery has considerable influence on the two performance criteria, and the MTCT algorithm is valuable for real-life workflow systems when both of the two optimization objectives are taking into account. However, being a multi-objective algorithm, the MTCT is inherently likely to diversify attention into other parameters. Kumar and Aramudhan [34] introduce a task scheduling technique in cloud computing using a hybridization of BA technique with gravitational scheduling algorithm taking into account deadline controls and trust model. The tasks are mapped to resources on the basis of trust level. The hybridized algorithm is experimented and proficiently minimizes the makespan and also the amount of failed tasks in comparison with GVSA. However, the BA is also known for weak local search when dealing with complex problems. An intelligent technique called NSGA-II is also presented to decipher the fault tolerance problem [24]. The NSGA-II technique is based on the Pareto dominance relationship, giving no distinct optimal result, but a set of results that are not subject on one another. The feat of the technique integrated with GA is demonstrated by a number experimental result. The average response time outcomes are highly interrelated with the makespan outcomes; still the general tendency is more complex to explain. Overall, the best outcomes are given by strategies favoring reliability. The related outcomes show the efficiency of the presented technique and GA for small- and medium-sized scheduling problems. However, the experimental results did not demonstrate the performance with large and massive cloud-based scheduling problems.

A fault tolerance and QoS scheduling based on content addressable network (CAN) in mobile social cloud computing (MSCC) is presented by [35]. CAN as the basic MSCC to carefully control the positions of mobile devices. An experimental simulation of the scheduling of both with and without CAN is presented. The simulation results show that the CAN fault tolerance scheduling algorithm enhances cloud service execution time, finish time and reliability to minimizes the cloud service error rate. However, the CAN technique is only tested in mobile cloud computing service and also there are no comparative results with any state-of-the-art intelligent algorithms. Tawfeek et al. [26] presents a dynamic cloud task scheduling policy based on ACO intelligent optimization technique. It is random optimization search approach that will be used for allocating the incoming tasks to the VMs. Simulation outcome shows that cloud task scheduling based on ACO surpasses FCFS and RR

algorithms. However, the dynamic scheduling in this technique is not a priority in achieving the goal of fault tolerance. Particle swarm optimization (PSO) schemes are also nature-inspired population-based intelligence techniques. The algorithms imitate the social characteristics of birds flocking and fishes schooling. By initiating a randomly dispersed set of particles which are called potential solutions, the scheme tries to develop solutions according to a quality measure which is called fitness function [36]. Yuan et al. [37] put forward a virtual machines scheduling scheme that takes into account the computing power of processing rudiments and also considers the computational density of the system. An improved PSO to address the VM scheduling problem in the cloud computing environment is presented. Verma and Kaushal [38] also present a bi-criteria priority-based particle swarm optimization (BPSO) to schedule workflow jobs in a given cloud computing environment for resources that reduced the execution cost and the execution time under a given deadline and capital. Similarly, the PSO have been adapted in grid and cloud scheduling to solve the problem of load balancing [39], service selection in grid [40], tunable workflow in cloud [41] and energy-aware tasks scheduling [42]. An energy-aware fault-tolerant scheduling (EAFTS) is put forward for public, multiuser cloud systems and investigates the three-way trade-off among reliability, performance and energy [43]. The technique consists of a static scheduling segment that runs on task graph using workload inputs before implementation, and a insubstantial dynamic scheduler that migrates processes for the duration of the implementation in case of undue re-executions. Experimental results show that compared to current VM or task replication methods, the proposed technique is capable of minimizing the overall application failure rates by over 50 % with about 76 % total energy overhead. However, the replication strategy used in this technique may affect the over system performance as well as the dynamic scheduling policy.

The current available resources in the cloud need to be applied at every scheduling point to avoid tasks failure due to VM failure or overloading. Current dynamic scheduling techniques did not either take fault tolerance parameters into account or are partially applied at different scheduling levels.

3 DCLCA fitness function

To derive the fitness function, consider that in cloud scheduling, the main goal of the providers is to reduce the completion time, while the aim of the clients is to reduce the price of accessing cloud resources by reducing the makespan time. Therefore, the fitness value of the DCLCA can be computed using the fitness function in Eq. 1

$$f(C) = \max\left\{\bigcup_{i=1}^{m} C_i\right\} \tag{1}$$

where C_i' is the completion of task i. The lesser the makespan the better the efficiency of the algorithm, meaning less time is taken to execute the algorithm.

The expected time of completion (ETC) is defined as the execution time for each task to compute on a certain VM obtained using the ETC matrix as shown in Eq. 2. Amount of tasks multiply by the amount of resources gives the dimension of the matrix, and its elements are represented as ETC(T_i, V_k). An ETC matrix related to this problem with n tasks $T = \{T_1, T_2,\ldots,T_n\}$ and m VMs represented as $V = \{V_1, V_2,\ldots,V_m\}$ resources

$$\mathrm{ETC} = T \cdot V = \begin{bmatrix} T_1V_1 & T_1V_2 & \cdot & \cdot & \cdot & T_1V_m \\ T_2V_1 & \cdot & & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot \\ T_nV_1 & \cdot & & \cdot & \cdot & T_nV_m \end{bmatrix} \tag{2}$$

Also, $P_f(j_i, r_k)$ is defined as the failure probability of running a task with security demand (SD) and a trust level (TL). The SD stands for the security demand for the application as at the time of submitting tasks. The superior is the SD value, the advance the security constraint for the application. The trust model appraises the VM site's reliability, to be precise, the TL. A task failure model is described as a function of the difference between the task's demand and machine's security. Equation 3 states the failure probability regarding a scheduling of a task T with a specific SD_j value, to the VM_i with trust value TL_i. TL stand for the security guarantee for the resources VM, the more is the TL value the more advanced the VM reliability [24, 44]

$$P_f(T_i, V_k) = \begin{cases} 0, & \text{if } \mathrm{SD}_i \le \mathrm{TL}_k \\ 1 - \mathrm{e}^{-\alpha(\mathrm{SD}_i - \mathrm{TL}_k)}, & \text{if } \mathrm{SD}_i > \mathrm{TL}_k \end{cases} \tag{3}$$

where α is the failure coefficient which is a fraction number.

4 Fault tolerance scheduling components

Figure 1 shows the scheduling components for the fault tolerance aware technique. We describe in detail functionalities and synchronous workings of each of the components in this section.

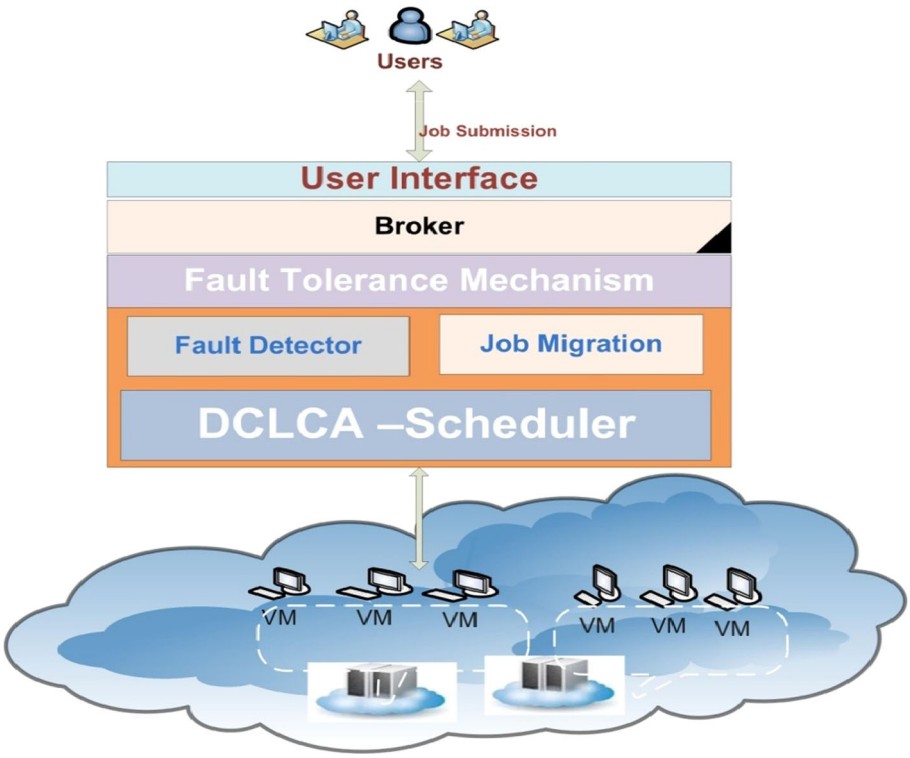

Fig. 1 Fault tolerance scheduling components

4.1 Fault detector

Fault detector is a necessity in designing fault tolerance mechanism. Many detection algorithms have diverse area of emphasis on specific parameters, for instance fault coverage, complexity and performance, etc. Previous fault detection techniques are taxonomized by system level pecking order which is also used in this proposed fault tolerance LCA scheduling technique for the failure discovery on the operating system level, VM level and also at the application level. In addition, VM introduced new facilities for fault detection [45].

The tasks execution running on a VM can be scrutinized from remote site and the tasks failure can be detected by the abnormal internal implementation information, like the abnormal cycles of system execution calls [46, 47]. VM detection can be achieved by executing a detection component located in virtual machine managers (VMMs) that intermittently judge the fault status of VM. One of the functions of the fault detector we present here traces the failed task or VM and then schedules healing sub-module in succession with a healing or recovery LCA scheduling algorithm. The healing module is to direct the resultant healing sub-modules to recover the faults according to fault intensity and category. The fault's healing is accomplished one after the other until the task is fully recovered.

4.2 Task migration

Task migration process involves the reassigning of jobs from the queues of faulty resources to the heads of queues of idle resources when those are accessible. It also solves the tasks fragmentation problem. Task migration algorithm reschedules abortive or failed tasks (T_n) to another available or under-loaded virtual machine VM_j known as its backup site. In case, some jobs did not complete execution on a particular VM due to some reasons (like job overloading or VM_i failure), the aborted or suspended jobs (T_n) can be instantly migrated to another VM_j for execution. Job migration increases resource utilization and also provides alternative resources in case of VM failure or overloading as shown in Fig. 2.

According to Rathore and Chana [48], task migration algorithms can be very helpful in solving the following issues during scheduling.

- Task migration algorithms are helpful in providing fault tolerance awareness when executing a long-running

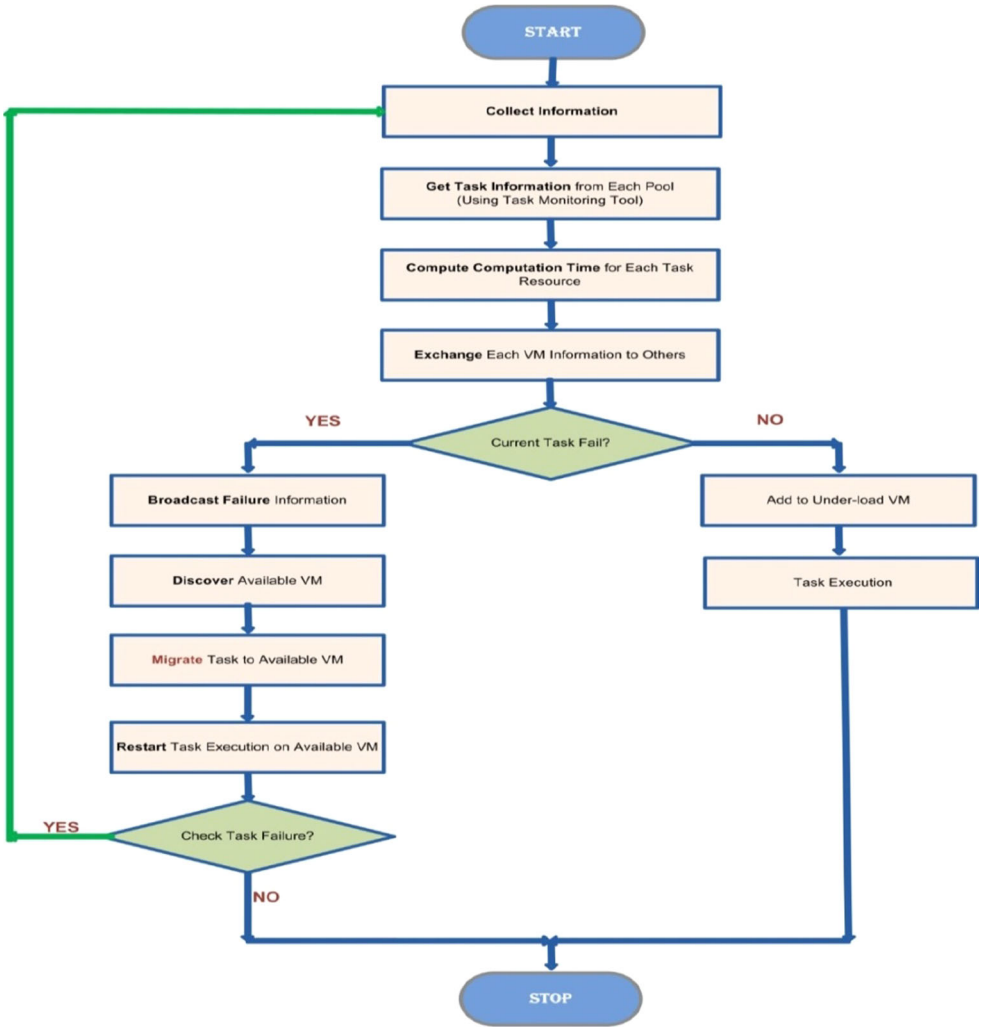

Fig. 2 Flowchart of task migration

tasks, VM failure, VM overload or system maintenance.

- Task migration algorithms can be very useful in handling the problem of load balancing in an over-loaded system. If a VM in a pool suddenly becomes overloaded, the whole tasks on that VM can be migrated to an under-loaded VM.
- Task migration can be motivated by resource request. For instance, tasks may require the use of massive databases, accessible on a devoted VM of the cloud.

5 Dynamic clustering league championship algorithm

The proposed dynamic clustering league championship algorithm (DCLCA) task scheduling technique is also designed by improving the LCA intelligent algorithm inspired by the analogy of sporting contests. The dynamic clustering algorithm is utilized to update and reflect the current status of the cloud VM resources as shown in Fig. 3.

5.1 Task clustering

The main purpose of task clustering is to allocate any cluster of task, to be executed on any accessible VMs dynamically. Figure 3 shows the dynamic task clustering steps designed to achieve DCLCA. Consider a subset of tasks $j_n \in J$, where P_n represents a partition $P_n = \{j_1, j_2, j_3, \ldots, j_n\}$ of J into n clusters. Therefore,

1. $j_i \neq \emptyset \quad i = 1, 2, 3, \ldots, n$
2. $j_i \cap j_j = \emptyset \quad i, j = 1, 2, 3, \ldots, n \; i \neq j$
3. $\bigcap_{i=1}^{n} j_i = J$

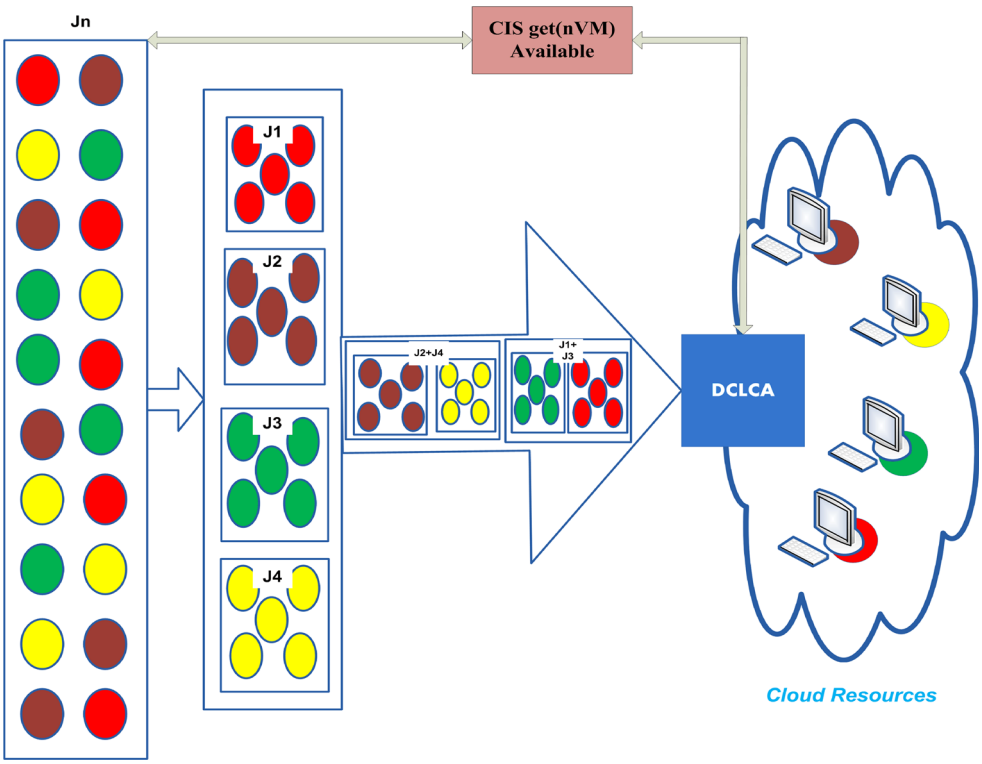

Fig. 3 Dynamic task clustering

This clustering step basically is meant to categorize the finest task-cluster to VM mapping using the cloud information system (CIS) to dynamically obtain the number of available virtual machines (nVM), anticipated execution time of cluster and capacity of the selected VM. Conversely, as a result of the dynamic characteristic of both tasks and resources, the volume of cluster is tentative for more resourceful usage. At each point of the scheduling, the current CIS information is used to determine the current number of available VM in order to dynamically partition the tasks in accordance with the current number of available resources. Algorithm 1 shows the dynamic clustering pseudo-code.

Algorithm 1. Dynamic Clustering

1. **Initialization**
2. **Get** n from the CIS
3. $P_n = \{j_1, j_2, j_3, \ldots, j_n\}$;
4. $j_i = \{J\} \quad i = 1, 2, 3, \ldots, n$;
5. $k = 0$
6. **Let** $P_n = P_{n-k}/(n-k)$
7. **Current step:**
8. **While** $(n-k<1)$ and $(P_n \neq \emptyset)$ **Do**
9. *select* $j_i\, j_j \in P_{n-k}$
10. $P_{n-k} = (P_{n-k} \cup j_i)/(n-k)$
11. k = k + 1
12. **Get** current n from CIS
13. **EndWhile**

Table 1 Parameters matching

LCA	EA	Remark
League L	Population	Generate from dataset
Week t	Iteration	Number of runs
Team i	ith member of the population	ith task to be executed
Formation X_i^t	Solution	Present best solution
Playing strength $f(X_i^t)$	Fitness function	Based on an objective function
Number of seasons S	Maximum iterations	Maximum schedules

5.2 Parameters matching

In order to achieve optimization with the proposed algorithm DCLCA in scheduling cloud tasks, we must first have to match the corresponding variables or parameters of the two systems. To achieve this, a simple comparison is made with the variables of a known evolutionary algorithm (EA) and the following matching is achieved (Table 1).

5.3 Winner/loser determination

One of the most important features of the LCA is the winner/loser determination technique [49]. In this research work, we utilized this feature in determining which cluster of tasks is scheduled on which VM in the IaaS cloud. In a cloud computing system, tasks sent by users are contest to get access to resources for effective scheduling and their best fitness value is evaluated on the basis of win/loss/tie for each of the tasks. For instance, in football league, each club is to get three points for a victory, zero for defeat and one for draw. By ignoring irregular abnormalities which may ensure even outstanding clubs in a variety of unsuccessful outcomes, it is probable that a more dominant club having a superior playing pattern defeats the lesser team. In an ideal league situation that is free from uncertainty effects, an assumption can be easily made for a linear correlation between the playing pattern of a club and the result of its matches. Utilizing the fitness function condition, the winner/loser decision in LCA is determined in a stochastic approach using criteria that the probability of winning or optimally scheduling tasks is relative to its degree of fitness value. Given cloud tasks i and j which are send to access cloud resources (VM) at a given time t, with the formations x_i^t and x_j^t and fitness functions (strength) $f(x_i^t)$ and $f(x_j^t)$, correspondingly. Let P_i^t represents the probability of tasks i to access the VM resources instead of task j at time t (P_i^t is defined, respectively). Given f be an ideal value.

$$\frac{f\left(x_i^t\right)-\hat{f}}{f\left(x_j^t\right)-\hat{f}}=\frac{p_i^t}{p_j^t} \tag{4}$$

From the LCA idealized rule, we deduce that

$$p_i^t+p_j^t=1 \tag{5}$$

From Eqs. 4 and 5 above, we solve for P_i^t

$$p_i^t=\frac{f\left(x_j^t\right)-\hat{f}}{f\left(x_j^t\right)+f(x_i^t)-2\hat{f}} \tag{6}$$

In order to find the winner or loser, a random number in between 0 to1 is generated; if the generated number is $\leq P_i^t$, it means task i won and task j lost, else j won and i lost. This method of finding the winner or loser is in line with the idealized rules. If by chance $f(x_i^t)$ approaches $f(x_j^t)$, then P_i^t can be arbitrarily approaching ½. But, if $f(x_j^t)$ becomes far $>f(x_i^t)$, also written as $f(x_j^t) \gg f(x_i^t)$, then P_i^t tends to one. Then, the value of f may be unobtainable in the feature, we use from the best function value found so far (that is, $\hat{f}^t=\min_{i=1,\ldots,L}\{f(B_i^t)\}$.

Using the strengths and weaknesses of each cluster of tasks, we created a good fitness value by taking different constraint into consideration. Likewise, a process is also carried out using artificial analysis method, which is SWOT (i.e., strengths, weaknesses, opportunities and threats) to generate an appropriate focus strategy. Considering that as a rule, cluster of tasks with their recent best fitness value, while planning the necessary changes suggested from the artificial analysis; the fresh solution $x_i^{t+1}=(x_{i1}^{t+1},x_{i2}^{t+1},\ldots,x_{in}^{t+1})$ for a cluster of tasks i where ranges from $i=1,\ldots,L$ at a time $t+1$ could be evaluated based on [50] as presented in Eq. 7

$$x_{id}^{t+1}=\begin{cases} b_{id}^t+y_{id}^t\left(\psi_1 r_{1id}\left(x_{id}^t-x_{kd}^t\right)+\psi_1 r_{2id}\left(x_{id}^t-x_{jd}^t\right)\right) & \text{if } f\left(x_i^t\right)>f\left(x_j^t\right)\cap f\left(x_i^t\right)>f\left(x_k^t\right) \\ b_{id}^t+y_{id}^t\left(\psi_2 r_{1id}\left(x_{kd}^t-x_{id}^t\right)+\psi_1 r_{2id}\left(x_{id}^t-x_{jd}^t\right)\right) & \text{if } f\left(x_i^t\right)>f\left(x_j^t\right)\cap f\left(x_k^t\right)>f\left(x_l^t\right) \\ b_{id}^t+y_{id}^t\left(\psi_1 r_{1id}\left(x_{id}^t-x_{kd}^t\right)+\psi_2 r_{1id}\left(x_{jd}^t-x_{id}^t\right)\right) & \text{if } f\left(x_j^t\right)>f\left(x_i^t\right)\cap f\left(x_l^t\right)>f\left(x_k^t\right) \\ b_{id}^t+y_{id}^t\left(\psi_2 r_{1id}\left(x_{kd}^t-x_{id}^t\right)+\psi_2 r_{2id}\left(x_{jd}^t-x_{id}^t\right)\right) & \text{if } f\left(x_j^t\right)>f\left(x_i^t\right)\cap f\left(x_k^t\right)>f\left(x_l^t\right) \end{cases} \tag{7}$$

The pseudo-code above shows that d is the dimension index. r_{1id} and r_{2id} are uniform random values between zero and one. ψ_1 and ψ_2 are coefficients that are used to measure the inputs of "retreat" or "approach" mechanisms, in that order. It is also important to note the distinct sign in parenthesis outcomes increase in the direction of the winner or retreat away from the loser. To generate a new schedule, a random number of changes made in B_i^t can be calculated using Eq. 8

$$q_i^t = \left[\frac{\ln\left(1-\left(1-(1-p_c)^{n-q_0+1}\right)r\right)}{\ln(1-p_c)}\right] + q_0 - 1,$$
$$q_i^t \in \{q_0, q_0+1, \ldots, n\} \tag{8}$$

where r represents the random number generated between zero to one and $p_c < 1, p_c \neq 0$ donating a controlling variable.

6 Experimental setup

To evaluate the proposed DCLCA fault-tolerant aware task scheduling technique, a cloud simulator has been used. The implementation and evaluation is done using the CloudSim 3.0.3 toolkit [51] on the Eclipse IDE Luna release 4.4.0. The simulation is done using two different scenarios. Task traces in the first scenario are generated from the Parallel Workload Archive [52] which contains 73,496 tasks. This workload archive is made available by San Diego Supercomputer Center (SDSC) and is in the standard workload format (SWF) recognized by the CloudSim. While the tasks traces in the second scenario are generated from the CloudSim's Workload PlanetLab. The DCLCA parameters are set at $\psi_1 = \psi_2 = 0.5$ and $p_c = 0.01$ which the selection of these values are based on [7].

Algorithm 2. Dynamic Clustering League Championship Algorithm (DCLCA)

1. **Start**
2. Obtain information about the list of task to be scheduled
3. **Initialize** population size L, number of initial maximum iterations S and set t=1
4. Obtain the number of available VMs from the CIS
5. **Generate** a present best solution for L-1 with initial iteration through each task formation x_i^t
6. **Generate** the tasks cluster dynamically using **Call** (*Algorithm 1*)
7. **Set** task cluster formations x_i^t and establish the fitness values for each cluster of task
8. **Let** the initialization be the tasks' present best solution
9. **While**[t <= S (L − 1])]
10. Using the task cluster schedule at t, find the winner/loser among each pair of tasks by utilizing the probability function $P_f(j_i, r_k)$ defined in *equation 3*
11. t=t+1
12. **For** i = 1 to L
13. Formulate a new optimal solution x_i^{t+1} for the next task using SWOT analysis in *equation 7*, while taking into consideration the task's new optimal solution $f(x_i^{t+1})$
14. **Compute** fitness values for the new system using *equation 1*
15. **If** x_i^{t+1} is fitter than x_i^t **Then** replace as new optimal solution
16. **End For**
17. **If** mod (t, L-1) = 0
18. **Check** fault
19. **If** fault = True **Then Call** (*Task Migration*) *//**Task Migration**//*
20. **Output** a task with best fitness value as final optimal solution
21. **Output** the smallest makespan using *equation 1*
22. **Output** task failure ratio using *equation 4*
23. **Output** task failure slowdown using *equation 5*
24. **End if**
25. **EndWhile**
26. **End**

Five different scheduling algorithms are used to compare the performance and effectiveness of our new proposed fault-tolerant scheduling technique based on the DCLCA. The techniques include: MTCT [33] and MAX-MIN [53], ACO [26] and the NSGA-II [24]. The ACO parametric values are set according to [26]; number of ants in colony = 10, evaporation factor $\rho = 0.4$, pheromone tracking weight $\alpha = 0.3$, heuristic information weight $\beta = 1$ and pheromone updating constant $Q = 100$. While the NSGA-II parameters are set according to [24]; population size = 1000, maximal iteration = 1000, the crossover rate = 0.5 and mutation rate = 0.1. The experiments are repeated ten times for each of the techniques and the averages of makespan time, failure ratio, failure slowdown and the performance improvement rate (in percentage) are observed. The two scenarios of the experiments are repeated with the same parametric values for all the chosen scheduling techniques.

6.1 Performance metric

To measure and compare the performance of the fault tolerance task scheduling mechanism in IaaS cloud computing environment, a number of performance parameters are considered. These includes the failure ratio, the failure slowdown and the performance improvement rate [54].

6.1.1 Makespan time

The makespan is the maximum completion time or the time when IaaS cloud system complete the latest task. So, if C_{ij} define the time that resource V_i needs to complete task T_j. Therefore, ΣC_i is the total time that resource V_i completes all the tasks submitted to it. Equation 1 defines the makespan in cloud environment mathematically.

6.1.2 Failure ratio

The failure ratio (FR) is the ratio of sum total of failed tasks in the proposed technique to the sum total of failed tasks in the other scheduling technique. The proposed LCA technique will be improved if the value of FR becomes less than one and is calculated using Eq. 12

$$\mathrm{FR} = \frac{\sum_0^n \text{no. of failures(DCLCA)}}{\sum_0^n \text{no. of failures(other scheme)}}. \tag{12}$$

6.1.3 Failure slowdown

Failure slowdown (FD) is described as the ratio of time delay or interruption as a result of failure-to-failure-free task execution time, mean over the sum total other tasks. The FD of the proposed DCLCA fault-tolerant task scheduling technique should be smaller than that of other scheduling techniques used for the comparison and is calculated using Eq. 13

$$\mathrm{FD} = \frac{\text{Time delay by failure to failure free task execution time}}{\text{Average total number of tasks}}. \tag{13}$$

6.1.4 Performance improvement rate

Performance improvement rate (PIR) is defined as the percentage of performance improvement (or reduction in makespan) for the proposed DCLCA technique with regard to the other techniques and is calculated using Eq. 14

$$\mathrm{PIR}(\%) = (\text{makespan(other scheme)} - \text{makespan(DCLCA)}) \times \frac{100}{\text{makespan(DCLCA)}}. \tag{14}$$

7 Results and discussion

This section presents and discusses the results of the experiments in the two formulated scenarios, so as to evaluate the efficiency of the proposed DCLCA technique.

7.1 First scenario

In the first scenario, five cloud users are created with five brokers and two data centers. The first data center contains three hosts, while the second data center contains two hosts. Ten VMs are also created using the Time_-Shared policy, each with 512 BM, image size of 10,000 BM, one CPU each, managed by Xen as the virtual machine manager (VMM) on Linux operating system. The host memory is 2048 MB, with storage capacity of 1,000,000 and a bandwidth of 10,000. Also, the number of tasks (cloudlets) submitted ranges between 10 and 100 each with a length of 800,000 and a file size of 600. Figures 4, 5 and 6 and Table 2 present the results obtained from this scenario.

From Fig. 4, it shows that makespan (completion time of the last task to be executed) increases as we increase the number of cloudlets in all the six techniques under consideration. When small number of tasks is sent for execution, all the algorithms return relatively similar makespan time with the DCLCA showing only slight improvement. As we continue to increase the number of tasks from 10 to 100, the makespan time of the algorithms keeps widening with the DCLCA returning less time. This means that,

DCLCA takes less time to execute the cloud tasks than the remaining algorithms under consideration. Table 2 shows that the DCLCA present a PIR % of 57.8, 53.6, 24.3 and 13.4 % over the MTCT, MINMAX, ACO and NSGA-II, respectively.

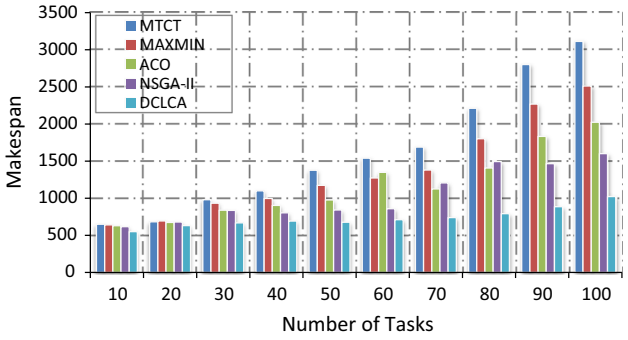

Fig. 4 Makespan time of first scenario

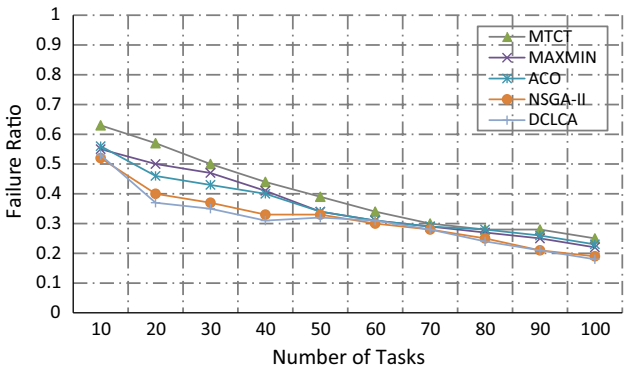

Fig. 5 Failure ratio of first scenario

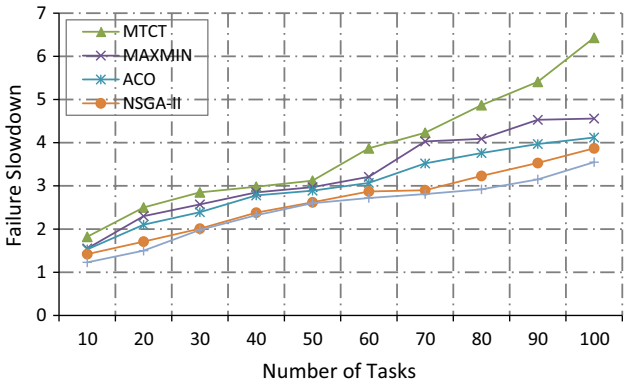

Fig. 6 Failure slowdown of first scenario

Figures 5 and 6 present the failure ratio and the failure slowdown, while Table 2 presents the performance improvement rate (in percentage), respectively. The failure ratio as compared to the nearest intelligent algorithms decreases as we increase the number of tasks for all the techniques. The lesser is the failure ratio, the better success of task execution rate. The DCLCA also shows improvement in the failure ratio as it returns lesser ration with increase in tasks and compared to the other techniques. The result obtained shows that failure slowdown of our proposed DCLCA technique is less than the other intelligent algorithms. It also shows that the PIR % of our proposed technique improved better and faster than the other algorithms as it relates to minimum makespan time and reliability. This means that the proposed DCLCA scheduling technique is more fault-tolerant and reliable than the other algorithms under consideration. The likely reason the DCLCA outperforms existing approaches under consideration is due to its dynamic nature to immediately reflect the current state of the resources in the heterogeneous environment. It can as well likely be as a result of its efficient performance in both local and global search as compared to other meta-heuristics.

7.2 Second scenario

The second scenario, we set 10 cloud users with 10 brokers and five data centers. Each of the data centers contains three hosts, making a total of 15 hosts. A total of 25 VMs are also created using the Space_Shared policy, each with 512 BM, image size of 20,000 BM, one CPU each, managed by Xen as the VMM on Linux operating system. The host memory is 2048 MB, with storage capacity of 1,000,000 and a bandwidth of 10,000. Also, the number of tasks (cloudlets) submitted ranges between 50 and 500 each with a length of 900,000 and a file size of 1000. Figures 7, 8 and 9 and also Table 3 present the results obtained from this scenario.

Similarly, in Fig. 7 it shows that when small number of tasks is sent for execution, all the algorithms return relatively similarly makespan time with the GA and DCLCA showing only slight improvement with lesser makespan

Table 2 DCLCA performance improvement rate (%) on makespan

	MTCT	MAXMIN	ACO	NSGA-II	DCLCA
Total makespan	14,042.7	13,671.9	11,057.4	10,099.5	8898.8
PIR % over MTCT		2.7	27.0	39.0	57.8
PIR % over MAXMIN			23.6	35.4	53.6
PIR % over ACO				9.5	24.3
PIR % over NSGA-II					13.5

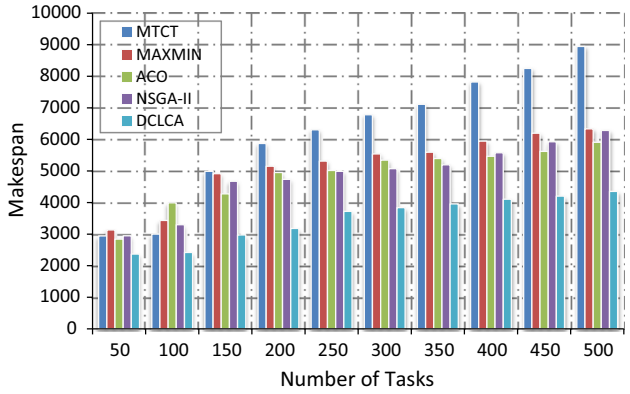

Fig. 7 Makespan time of second scenario

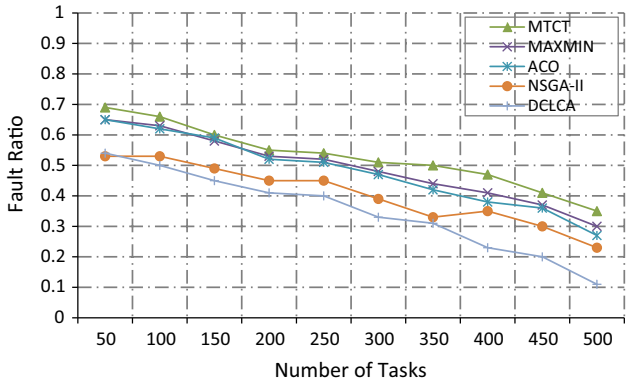

Fig. 8 Failure ratio of second scenario

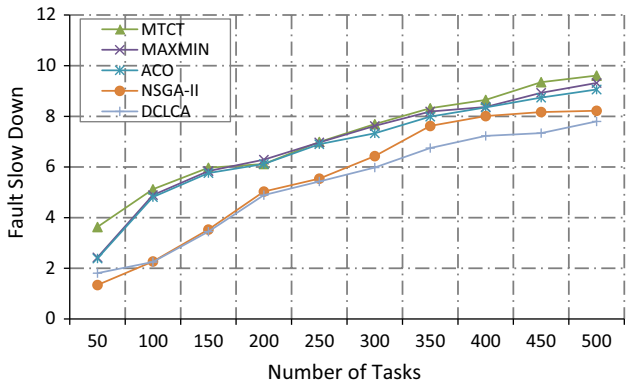

Fig. 9 Failure slowdown of second scenario

time. When the number of tasks increase from 50 to as much as 500, the makespan time of the algorithms keeps widening with the DCLCA returning lesser time. Table 3 shows that the DCLCA present a PIR % of 60, 38.9, 31.5 and 31.2 % over the MTCT, MINMAX, ACO and NSGA-II, respectively. This shows that the DCLCA use lesser time to implement the cloud tasks than the remaining algorithms under consideration.

In Figs. 7, 8 and 9 and Table 3, the DCLCA shows improvement in terms of failure ration as it returns lesser ration with increase in tasks in comparison to the other algorithms. The result obtained implies that failure slowdown of our proposed DCLCA technique is less than the other intelligent algorithms. Simulation evaluation also shows that the performance improvement rate of our proposed technique improved better and more speedily than the other algorithms as relates to minimum makespan time and reliability. This shows that the proposed fault-tolerant aware DCLCA scheduling technique performance and reliable than the other algorithms under consideration.

The proposed DCLCA approach is valuable for the management dynamic and responsive faults by predicting tasks failures along with scheduling. By implication, it explores the prospects for failure projection and handling in cloud applications so as to minimize the depletion of resources by tasks that fail in the process of execution. This study is also important because of the performance variations experienced among cloud resources and the imminent occurrence of failure during application scheduling, environment and the network. The contributions of this study help in making the execution of tasks faultless on cloud computing systems.

8 Summary and conclusion

We proposed a DCLCA technique for dynamic clustering fault tolerance aware intelligent scheduling using the LCA optimization algorithm. Task migration and fault detector strategies are also implemented as additional fault reduction components with an efficient method of scheduling in order to minimize makespan time. In order to evaluate the proposed technique, the CloudSim simulation toolkit is

Table 3 DCLCA Performance improvement rate (%) on makespan

	MTCT	MAXMIN	ACO	NSGA-II	DCLCA
Total makespan	59,411.3	51,571.4	48,821.0	48,714.5	37,131.4
PIR % over MTCT		15.2	21.7	22.0	60.0
PIR % over MAXMIN			5.6	5.9	38.9
PIR % over ACO				0.29	31.5
PIR % over NSGA-II					31.2

used to create two different scenarios in IaaS cloud environment using two different datasets.

The result of our experiment shows that the proposed DCLCA intelligent technique returned a significant fault reduction in task failure as measured in terms of failure ratio, failure slowdown and PIR parameters. It also proves that the proposed DCLCA technique performs better than the MTCT, MAXMIN, ACO and NSGA-II by returning reduced makespan time in addition to the above mentioned fault tolerance parameters in the two different simulated scenarios. In view of these experimental results, it shows that our proposed DCLCA fault aware technique provides better quality scheduling results than the other intelligent techniques. This indicates that the technique is very appropriate for task execution in the cloud computing environment. Therefore, the authors would wish to recommend hybridization of the LCA with other effective intelligent scheduling techniques in other to produce more performance in terms of fault tolerance. We also wish to recommend testing the technique in a real cloud environment.

Acknowledgments The authors would like to acknowledge and appreciate the support of Universiti Teknologi Malaysia (UTM), Research University Grant Q. J130000.2528.05H87 and the Nigerian Tertiary Education Trust Fund (TetFund) for their support.

References

1. Gital AY, Ismail AS, Chen M, Chiroma H (2014) A framework for the design of cloud based collaborative virtual environment architecture. In: Proceedings of the international multi conference of engineers and computer scientists
2. Lu K, Yahyapour R, Wieder P, Yaqub E, Abdullah M, Schloer B, Kotsokalis C (2016) Fault-tolerant service level agreement lifecycle management in clouds using actor system. Future Gener Comput Syst 54:247–259
3. Moon Y-H, Youn C-H (2015) Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks. Comput Netw 82:81–95
4. He J, Dong M, Ota K, Fan M, Wang G (2014) NetSecCC: a scalable and fault-tolerant architecture for cloud computing security. Peer-to-Peer Netw Appl 9(1):67–81
5. Nawi NM, Khan A, Rehman MZ, Chiroma H, Herawan T (2015) Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification. Math Probl Eng. doi:10.1155/2015/868375
6. Mills B, Znati T, Melhem R (2014) Shadow computing: an energy-aware fault tolerant computing model. In: 2014 International conference on computing, networking and communications (ICNC), IEEE, pp 73–77
7. Kashan HA (2009) League championship algorithm: a new algorithm for numerical function optimization. In: International conference of soft computing and pattern recognition, 2009. SOCPAR'09, IEEE, pp 43–48
8. Kashan HA, Karimi B (2012) A new algorithm for constrained optimization inspired by the sport league championships. In: 2010 IEEE congress on evolutionary computation (CEC), IEEE, pp 1–8
9. Abdulhamid SM, Latiff MSA, Ismaila I (2014) Tasks scheduling technique using league championship algorithm for makespan minimization in IAAS cloud. ARPN J Eng Appl Sci 9(12):2528–2533
10. Abdulhamid SM, Latiff MSA, Madni SHH, Oluwafemi O (2015) A survey of league championship algorithm: prospects and challenges. Indian Jo Sci Technol 8(S3):101–110
11. Yang Y-G, Tian J, Lei H, Zhou Y-H, Shi W-M (2016) Novel quantum image encryption using one-dimensional quantum cellular automata. Inf Sci 345:257–270
12. Dondi R, El-Mabrouk N, Swenson KM (2014) Gene tree correction for reconciliation and species tree inference: complexity and algorithms. J Discrete Algorithms 25:51–65
13. Abdulhamid SM, Latiff MSA, Bashir MB (2014) On-demand grid provisioning using cloud infrastructures and related virtualization tools: a survey and taxonomy. Int J Adv Stud Comput Sci Eng IJASCSE 3(1):49–59
14. Kushwah VS, Goyal SK, Narwariya P (2014) A survey on various fault tolerant approaches for cloud environment during load balancing. Int J Comput Netw Wirel Mobile Commun 4(6):25–34
15. Yang W, Zhang C, Shao Y, Shi Y, Li H, Khan M, Hussain F, Khan I, Cui L-J, He H (2014) A hybrid particle swarm optimization algorithm for service selection problem in the cloud. Int J Grid Distrib Comput 7(4):1–10
16. Hussin M, Lee YC, Zomaya AY (2010) Dynamic job-clustering with different computing priorities for computational resource allocation. In: Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, IEEE Computer Society, pp 589–590
17. Vidhate D, Patil A, Guleria D (2010) Dynamic cluster resource allocations for jobs with known memory demands. In: Proceedings of the international conference and workshop on emerging trends in technology, ACM, pp 64–69
18. SiM Abdulhamid, Latiff SMA, Bashir MB (2014) Scheduling techniques in on-demand grid as a service cloud: a review. J Theor Appl Inf Technol 63(1):10–19
19. Abdullahi M, Ngadi MA (2016) Symbiotic organism search optimization based task scheduling in cloud computing environment. Future Gener Comput Syst 56:640–650
20. Madni SHH, Latiff MSA, Coulibaly Y (2016) An appraisal of meta-heuristic resource allocation techniques for IaaS cloud. Indian J Sci Technol 9(4):1–14. doi:10.17485/ijst/2016/v9i4/80561
21. Chiroma H, Shuib NLM, Muaz SA, Abubakar AI, Ila LB, Maitama JZ (2015) A review of the applications of bio-inspired flower pollination algorithm. Procedia Comput Sci 62:435–441
22. Boru D, Kliazovich D, Granelli F, Bouvry P, Zomaya AY (2015) Energy-efficient data replication in cloud computing datacenters. Clust Comput 18(1):385–402. doi:10.1007/s10586-014-0404-x
23. Gangeshwari R, Subbiah J, Malathy K, Miriam D (2012) HPCLOUD: a novel fault tolerant architectural model for hierarchical MapReduce. In: 2012 international conference on recent trends in information technology (ICRTIT), IEEE, pp 179–184
24. G$_{,}$*asior* J, Seredyński F (2013) Multi-objective parallel machines scheduling for fault-tolerant cloud systems. In: Joanna K, Di Martino B, Talia D, Xiong K (eds) Algorithms and architectures for parallel processing. Springer, Switzerland, pp 247–256. doi:10.1007/978-3-319-03859-9_21
25. Ling Y, Ouyang Y, Luo Z (2012) A novel fault-tolerant scheduling algorithm with high reliability in cloud computing systems. J Converg Inf Technol 7(15):107–115. doi:10.4156/jcit.vol7.issue15.13
26. Tawfeek M, El-Sisi A, Keshk A, Torkey F (2015) Cloud task scheduling based on ant colony optimization. Int Arab J Inf Technol (IAJIT) 12(2):129–137

27. Ganga K, Karthik S (2013) A fault tolerent approach in scientific workflow systems based on cloud computing. In: 2013 international conference on pattern recognition, informatics and mobile engineering (PRIME), IEEE, pp 387–390
28. Bala A, Chana I (2015) Autonomic fault tolerant scheduling approach for scientific workflows in Cloud computing. Concurr Eng 23(1):27–39. doi:10.1177/1063293X14567783
29. Bonvin N, Papaioannou TG, Aberer K (2010) A self-organized, fault-tolerant and scalable replication scheme for cloud storage. In: Proceedings of the 1st ACM symposium on cloud computing, ACM, pp 205–216
30. Sampaio AM, Barbosa JG (2015) A performance enforcing mechanism for energy-and failure-aware cloud systems. In: 2014 international green computing conference, IGCC 2014. doi:10.1109/IGCC.2014.7039151
31. Patra PK, Singh H, Singh G (2013) Fault tolerance techniques and comparative implementation in cloud computing. Int J Comput Appl 64(14):37–41
32. Nawi NM, Khan A, Rehman M, Chiroma H, Herawan T (2015) Weight optimization in recurrent neural networks with hybrid metaheuristic Cuckoo search techniques for data classification. Math Probl Eng 501:868375
33. Xu H, Yang B, Qi W, Ahene E (2016) A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery. KSII Trans Internet Inf Syst 10(3):976–995. doi:10.3837/tiis.2016.03.002
34. Kumar VS, Aramudhan M (2014) Hybrid optimized list scheduling and trust based resource selection in cloud computing. J Theor Appl Inf Technol 69(3):434–442
35. Choi S, Chung K, Yu H (2014) Fault tolerance and QoS scheduling using CAN in mobile social cloud computing. Clust Comput 17(3):911–926
36. Kaveh A (2014) Particle swarm optimization. In: Advances in metaheuristic algorithms for optimal design of structures. Springer, Switzerland, pp 9–40. doi:10.1007/978-3-319-05549-7
37. Yuan H, Li C, Du M (2014) Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing. J Softw 9(3):705–708
38. Verma A, Kaushal S (2014) Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. In: 2014 recent advances in engineering and computational sciences (RAECS), IEEE, pp 1–6
39. Ramezani F, Lu J, Hussain FK (2014) Task-based system load balancing in cloud computing using particle swarm optimization. Int J Parallel Program 42(5):739–754
40. Yang W, Zhang C, Shao Y, Shi Y, Li H, Khan M, Hussain F, Khan I, Cui L-J, He H (2014) A hybrid particle swarm optimization algorithm for service selection problem in the cloud. Int J Grid Distrib Comput 7(4):1–10. doi:10.14257/ijgdc.2014.7.4.01
41. Wu K (2014) A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing. Master's Projects, Paper 358. San José State University, USA
42. Zhang W, Xie H, Cao B, Cheng AM (2014) Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm. Math Probl Eng 2014: 1–9. doi:10.1155/2014/287475
43. Gao Y, Gupta SK, Wang Y, Pedram M (2014) An energy-aware fault tolerant scheduling framework for soft error resilient cloud computing systems. In: Design, automation and test in Europe conference and exhibition (DATE), 2014, IEEE, pp 1–6
44. Hu Y, Gong B, Wang F (2010) Cloud model-based security-aware and fault-tolerant job scheduling for computing grid. In: ChinaGrid conference (ChinaGrid), 2010 fifth annual, IEEE, pp 25–30
45. Qiang W, Jiang C, Ran L, Zou D, Jin H (2015) CDMCR: multi-level fault-tolerant system for distributed applications in cloud. Secur Commun Netw 2015:SCN-SI-077. doi:10.1002/sec.1187
46. Urgaonkar R, Wang SQ, He T, Zafer M, Chan K, Leung KK (2015) Dynamic service migration and workload scheduling in edge-clouds. Perform Eval 91:205–228. doi:10.1016/j.peva.2015.06.013
47. Vobugari S, Somayajulu D, Subaraya BM (2015) Dynamic replication algorithm for data replication to improve system availability: a performance engineering approach. IETE J Res 61(2):132–141. doi:10.1080/03772063.2014.988757
48. Rathore N, Chana I (2014) Load balancing and job migration techniques in grid: a survey of recent trends. Wirel Pers Commun 79(3):2089–2125
49. Yadav S, Nanda SJ (2015) League championship algorithm for clustering. In: 2015 IEEE power, communication and information technology conference (PCITC), IEEE, pp 321–326
50. Xu W, Wang R, Yang J (2015) An improved league champi-onship algorithm with free search and its application on pro-duction scheduling. J Intell Manuf 1–10. doi:10.1007/s10845-015-1099-4
51. Buyya R, Ranjan R, Calheiros RN (2009) Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities. In: International conference on high performance computing & simulation, 2009. HPCS'09, IEEE, pp 1–11
52. Parallel Workload Archive - SDSC-SP2-1998-4.swf (2015). http://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_sp2/index.html. Accessed 30 Jan 2015
53. Ramakrishnan L, Reed DA (2008) Performability modeling for scheduling and fault tolerance strategies for scientific workflows. In: Proceedings of the 17th international symposium on High performance distributed computing, ACM, pp 23–34
54. Garg R, Singh AK (2014) Fault tolerant task scheduling on computational grid using checkpointing under transient faults. Arabian J Sci Eng 39(12):8775–8791