Fault Tolerance Based Load Balancing Approach for Web Resources in Cloud Environment

Anju Shukla, Shishir Kumar, and Harikesh Singh
Department of Computer Science and Engineering, Jaypee University of Engineering and Technology, India

Abstract: *Cloud computing consists group of heterogeneous resources scattered around the world connected through the network. Since high performance computing is strongly interlinked with geographically distributed service to interact with each other in wide area network, Cloud computing makes the architecture consistent, low-cost, and well-suited with concurrent services. This paper presents a fault tolerance load balancing technique based on resource load and fault index value. The proposed technique works in two phases: resource selection and task execution. The resource selection phase selects the suitable resource for task execution. A resource with least resource load and fault index value is selected for task execution. Further task execution phase sets checkpoints at various intervals for saving the task state periodically. The checkpoints are set at various intervals based on resource fault index. When a task is executed on a resource, fault index value of selected resource is updated accordingly. This reduces the checkpoint overhead by avoiding unnecessary placements of checkpoints. The proposed model is validated on CloudSim and provides improved performance in terms of response time, makespan, throughput and checkpoint overhead in comparison to other state-of-the-art methods.*

Keywords: *Scheduler, checkpoint manager, cloud computing, checkpointing, fault index, high performance computing.*

Received June 6, 2018; accepted July 2, 2019
https://doi.org/10.34028/iajit/17/2/10

1. Introduction

Cloud computing provides collaboration and sharing of resources especially in three major forms: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [15]. Technologies, grid and cloud differ from each other in terms of architecture, applications and uses as shown in Table 1. Cloud users have their own private resource provided by the service provider while grid computing follows the distributed architecture in which task is assigned to remote resources for completing its execution. Due to distributed nature of applications in cloud environment, developer must deal with several issues like load balancing, access control, fault occurrence, communication and task scheduling [6].

Load balancing is the technique to distribute the load among servers optimally. The key concern of optimization is to minimize response time, execution time, overhead, and maximize throughput. Broadly, load balancing techniques can be classified in two categories [6]. Static Load Balancing (SLB) and Dynamic Load Balancing (DLB), SLB algorithm uses earlier information of the network state for assigning task to any resource while DLB distributes the workload at runtime among the available resources.

Sometimes when tasks executes on a resource, fault occurs and process doesn't finished due to several reasons: voluntarily leave or join characteristic of cloud resources, resource heterogeneity, interactive parallel applications, resource sharing etc., these situations are managed through fault tolerance techniques to provide the estimated quality results. Fault tolerance is the characteristic that allows the system to continue properly when fault occurs [3]. Two major techniques i.e., job replication and checkpointing are used to deal with fault conditions. In job replication, several copies of the task make available on different resources. In cloud environment, task is referred as cloudlet associated with length, input-output files, id, deadline etc., the major drawback of this technique is that task executes from the beginning on another resource, it increased total execution time of tasks. In checkpointing, the state of task saves periodically to avoid task execution from the very beginning. Many check pointing based fault tolerance techniques face several limitations like- analysis of checkpoint interval, resource selection, communication and checkpoint overhead etc.

Table 1. Comparison between grid computing and cloud computing.

Criteria	Grid computing	Cloud computing
Architecture	Distributed architecture	Client server architecture
Application type	Batch applications	Interactive applications
Reachability	Decentralized	Centralized
Uses	Describe large volume of data and information	Used to store data and information on remote servers
Service provider	Research institutes and universities organize their service around the world	Individual companies
Resource Management	Managed by providers and users	Managed by cloud providers only
Technology	Open source	Proprietary

Most of the existing work considers resource load as a main criteria to select a resource. Due to these limitations, this paper introduces a fault tolerance based load balancing approach for distributed tasks in cloud environment. Proposed algorithm selects the suitable resource based on Resource load (Resi-load) and Fault Index Value (FI-Val) for cloudlet execution.

2. Related Work

Load balancing has concerned the interest of researchers in the last decade. Various load balancing techniques have been designed to enhance the efficiency of the high performance computing framework [12, 13, 14, 15, 20].

Singh and Kumar [19] presented the Web Server Queueing (WSQ) algorithm for load balancing in distributed environment. Presented model is compared with Remaining Capacity (RC) and server content based queue (QSC) algorithms, and performs better in both homogeneous and heterogeneous environment.

Hao *et al.* [7] presented a load balancing mechanism based on deadline control. Resource broker analyzed the existing load for categorizing the resources in normal loaded, under loaded and overloaded resource list. Prediction of change in state from under loaded or normal loaded to overload is also done by calculating the surplus capacity of resources.

Patel *et al.* [17] presented modified load balancing algorithm presented by Hao *et al.* [7] for recovery from deadline failure. When a gridlet doesn't finish in deadline, result of partial executed gridlet is saved and resubmitted to other suitable resource to reduce execution time of gridlets.

Pao and Chen [16] proposed dispatching algorithm for load balancing of web server. The server with less capacity always serves fewer requests because it does not have sufficient processing power. The presented architecture uses remaining capacity of Domain Name System (DNS) server and mail server to manage the load in distributed system.

Arabnejad and Barbosa [1] presented a budget based task scheduling algorithm for load balancing. Algorithm works in two phases: task selection phase and processor selection phase. For selecting the task, priority is assigned by computing the rank. For balancing the load, worthiness of all processor is calculated and selects the processor with highest worthiness value.

Garg and Singh [5] presented a task scheduling algorithm to deal with faults. For improving execution time of tasks, Computation for fault tolerance and recovery overhead is done. Resources capacity is recalculated based on genetic algorithm in presence of failures. Result shows reduced execution time and improved task reliability.

Various authors focused on predicting execution time for optimize various performance metrics [2, 11]. Chang *et al.* [4] presented a resource selection based task scheduling algorithm for load balancing. Task scheduler is in-charge of transmitting tasks to resources depending on scheduling algorithm. Resources are categorized into L discrete levels (r1, r2….rl) from smallest to largest. Each task has a resource requirement Ri and a closeness factor (0<C<1). The fittest resource is selected based on Predicted Execution Time (PET) which results in reduced makespan and increased system throughput.

Various authors suggested solutions for adaptive load balancing in dynamic environment [8, 9, 10, 18]. Lee *et al.* [10] presented two scheduling algorithms to perform adaptive load balancing between clusters. Two algorithms differ with each other by mechanism of cluster selection. Balance threshold is used to adapt the changes of environment when load changes. The major contribution of work is load balancing and reduction in makespan.

To overcome these issues, a fault tolerance based load balancing approach for high performance computing is proposed in cloud environment. In the proposed approach, a least loaded and fault index value resource is selected for load balancing. The major advantage of the proposed approach over the other state-of-the-art methods include reduction in response time, makespan, communication and checkpoint overhead, increase in system throughput.

2.1. Contributions

Most of the research work of dynamic load balancing is done by optimizing metrics like- resource capacity, resource utilization, drop rate, resource cost, resource selection criteria, queueing analysis, deadline control and recovery. Due to dynamic nature of cloud environment, fault tolerance is an essential metric of concern for concurrent and distributed applications. Unfinished tasks need to transfer to another suitable resource for completing execution. A mechanism is required for fault notification and further execution of partial executed task. The major contributions of the proposed work are as follows:

- A fault tolerance model is proposed for partially executed tasks (cloudlets) due to resource failure based on checkpoint.
- The fault tolerance model proposed a mechanism to determine checkpoint interval to reduce overall execution time and checkpoint overhead of cloudlet.
- A resource selection mechanism is also proposed for cloudlet execution. As in literature [4, 7, 18, 19], only Resi-load should not be the criteria to select a resource for rescheduling of cloudlet. There may be a possibility that under loaded resource may also lead to execution failure. With this perspective, an algorithm is proposed to update the indexing of resource based on Resi-load and FI-Val to avoid the fault in the future scenario.

3. Proposed Fault Tolerance Based Load Balancing Model

Figure 1 shows the architecture of proposed fault tolerance model. The proposed algorithm is an improved version of Enhanced Gridsim with Deadline Control (EGDC) [7] and Enhanced GridSim with Load Balancing Based on Deadline Failure Recovery (EGDFR) [17] briefly explained in section related work. The proposed model works in two phases: resource selection and task execution. In resource selection phase, user submits cloudlet to cloud broker with deadline constraint. Cloud broker submits cloudlet to scheduler to select the suitable resource for execution. Two factors are considered for resource selection Resi-load and FI-Val. A resource with least Resi-load and FI-Val is selected for cloudlet execution.

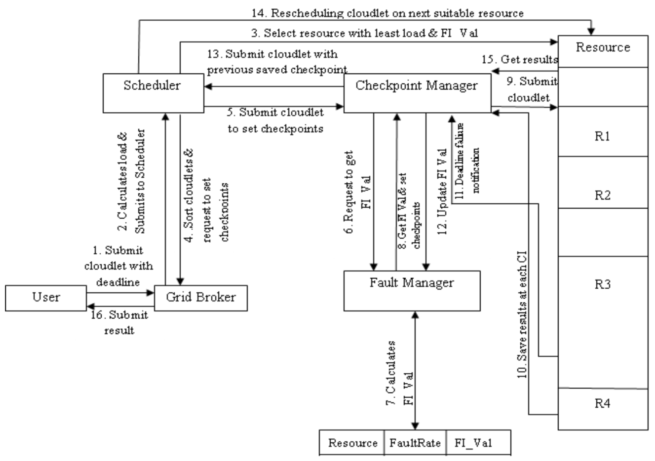
14. Rescheduling cloudlet on next suitable resource
3. Select resource with least load & FI Val
13. Submit cloudlet with previous saved checkpoint
15. Get results
Resource
Scheduler
Checkpoint Manager
9. Submit cloudlet
5. Submit cloudlet to set checkpoints
R1
R2
R3
R4
2. Calculates load & Submits to Scheduler
4. Sort cloudlets & request to set checkpoints
6. Request to get FI Val
8. Get FI Val & set checkpoints
12. Update FI Val
11. Deadline failure notification
1. Submit cloudlet with deadline
User
Grid Broker
16. Submit result
Fault Manager
10. Save results at each CI
7. Calculates FI Val
Resource
FaultRate
FI_Val

Figure 1. Proposed fault tolerance load balancing model.

To provide fault tolerance load balancing, intermediate results of executing cloudlets are saved at various intervals. Checkpoint intervals are decided based on FI-Val of a resource. Checkpoint manager gets the FI-Val of resource from fault manager and sets checkpoints accordingly. At each cloudlet execution, fault value of resource is updated. Checkpoint manager submits job to resource for execution. If deadline failure occurs, execution starts from the previous successful saved checkpoint. The uniqueness of proposed model is that checkpoints are set at the time of cloudlet scheduling rather than at the time of cloudlet execution.

3.1. Algorithm

Various notations and terminologies are used in proposed algorithm that is listed in Table 2. The detailed description of the proposed approach is presented here:

- *Step* 1: user submits tasks to cloud broker with deadline constraint defined by user. Broker determines load due to received cloudlet form Equation (1):

$$\text{load}_i = \text{length}/\text{deadline} \tag{1}$$

- *Step* 2: each resource has some initial load. Due to incoming cloudlet, resource state may become underloaded, overloaded or normalloaded. Scheduler finds the suitable resource by performing two tasks:
    - Arrange the underloaded resource list in ascending order of Res_i-load.
    - Arrange the sorted underloaded list in ascending order of FI-Val of resource.

After executing these two steps scheduler gets the resource with least Res_i-load and FI-Val. Load of selected resource is updated by using Equations (2), (3), and (4) respectively.

Table 2. Notations and Terminologies.

Parameters	Definitions
Cloudlet	Tasks associated with length, input-output files, id, deadline etc.,
ct	Cloudlet execution time
$\text{Res}_{i}\_\text{load}$	Load of ith resource
Rb	Underloaded threshold value
Rh	Normalloaded threshold value
PE	Number of CPU units (in MIPS)
M	Number of PE's
C_i	Capacity of ith resource
CPE	Available computing speed (MIPS)
FI_Val	Fault index value of resource
F_cloudlet	Finished cloudlet
U_cloudlet	Unfinished cloudlet
Chk_t	Checkpoint time
Chk_ID	Checkpoint ID
MI	Million Instructions
TET	Total Execution Time
N	Number of checkpoints
Chk_oh	Checkpoint overhead

$$Ci = \sum_{i=1}^{m} PEi * CPEi \tag{2}$$

$$\text{Factor}_i = \text{load}_i / C_i; \tag{3}$$

$$\text{Res}_{i}\_\text{load} = \text{Res}_{i}\_\text{load} + \text{Factor}_i \tag{4}$$

After load updation, $\text{Res}_{i}\_\text{load}$ is compared with rb and rh respectively. If selected resource is under loaded or normal loaded, then received cloudlet is assigned to selected resource otherwise cloudlet is added in unassigned cloudlet list for selecting another suitable resource and resource is added into overloaded resource list. Algorithm 1 shows selection and updation load of selected resource.

- *Step* 3: scheduler selects the resource that has least load and FI-Val for cloudlet execution.
- *Step* 4: scheduler arranges all cloudlets in ascending order of cloudlet length as shown in Algorithm 2
- *Step* 5: scheduler requests to checkpoint manager to set checkpoints at various intervals determined by FI-Val of the selected resource.

Algorithm 1. Resource selection and updation algorithm

```
Input: A list of underloaded resource (underloadedlist [i]), $Res_i$_load, $factor_i$
Output: Resource with least $Res_i$_load and FI_Val
BEGIN
1.  for (all underloaded list) do
2.      Arrange list in ascending order of $Res_i$_load
3.  end for
4.  for (all underloaded list) do
5.      Arrange list in ascending order of FI_Val
6.  end for
7.  for (all underloaded list) do
8.      //Select the first resource of list and update load due
        to incoming cloudlet
9.      $Res_i$_load= $Res_i$_load+$factor_i$
10. end for
11. for (all underloaded list) do
12.     //Check state of selected resource
13.     checkstate (resource r)
14.     {
15.             if($Res_i$_load <rb)
16.                     underloadedlist[i]);
17.             else
18.             if ( $Res_i$_load >rb) && ( $Res_i$_load <rh)
19.                     normalloadedlist[i]);
20.             else
21.             if ($Res_i$_load>=rh)
22.                     add resource in overloadedlist
23.             end if
24.             end if
25.             end if
26. end for
END
```

- *Step* 6: checkpoint manager requests to fault manager to provide the FI-Val of selected resource.
- *Step* 7: fault manager maintains F-cloudlets and U-cloudlets values for finished and unfinished cloudlets respectively. Fault Rate (F-Rate) is calculated using Equation (5). Algorithm 3 shows computation of FI-Val of selected resource.

$$\text{F\_Rate} = \frac{\text{U\_cloudlet}}{\text{F\_ cloudlet+U\_ cloudlet}} * 100 \quad (5)$$

Algorithm 2. cloudlett sorting algorithm

```
Input: List of unassigned cloudlet (unasignedcloudletlist [])
Output: sorted unassignedcloudletlist []
BEGIN
1.  for (all cloudletlength list) do
2.      for(unassignedcloudletlist) do
3.          Arrange cloudlets in ascending order of length
4.      end for
5.  end for
END
```

- *Step* 8: checkpoint manager gets the FI-Val of resource from fault manager. If FI-Val of resource is one then check point manager will set the checkpoints at intervals of 1ms. Each checkpoint is assigned a checkpoint number that is incremented at each interval by 1ms.
- *Step* 9: the cloudlet is submitted to selected resource with determined checkpoint interval.

Algorithm 3. Fault Index Value (FI_Val) Algorithm

```
Input: A list of underloaded resource (underloadedlist []),
F_Rate
Output: underloaded resource list (underloadedlist []) with
FI_Val
BEGIN
1.  for (underloadedlist) do
2.      if (F_Rate >=1) && (F_Rate< 10)
3.      FI_Val=1;
4.      end if
5.      else
6.      if (F_Rate >=10) && (F_Rate< 20)
7.      FI_Val=2;
8.              .
9.      ..
10.     else
11.     if (F_Rate>=90) && (F_Rate)<=100)
12.     FI_Val=10;
13.     end if
14.     end if
15.     end if
16. end for
END
```

- *Step* 10: checkpoint manager saves the intermediate results at each checkpoint interval and gets the result of executed cloudlet.
- *Step* 11: If deadline failure occurs then checkpoint manager gets notification from resource.
- *Step* 12: checkpoint manager informs to fault manager to update the FI-Val and resubmits partial executed cloudlet to scheduler with most recent saved checkpoint.
- *Step* 13: scheduler resubmits cloudlet to next suitable resource and recovery algorithm is executed to resume cloudlet execution.
- *Step* 14: scheduler waits for in-transit messages before starts execution on next underloaded resource. When all messages are recovered, then cloudlet resumes for execution.
- *Step* 15: based on execution status of cloudlet, Scheduler receives the results that are displayed to the user.

4. Simulation and Results

To verify the effectiveness of the proposed model, work is simulated on CloudSim simulator and results are compared with EGDC and EGDFR. Makespan, Average Response Time (ART), throughput, communication overhead and Checkpoint overhead (Chk-oh) are the major metrics considered for comparison. Simulation is performed using windows 7 on Intel Pentium (B940/2 GHz) with 4 GB Random Access Memory (RAM) and 500 Mega Byte (MB). We have considered two cases for simulation of proposed algorithm.

- *Case* 1: simulation by varying no. of cloudlets.
- *Case* 2: simulation by varying number of PEs.

4.1. Case 1-Simulation By Varying Number of Cloudlets

In this case, simulation is performed by varying number of cloudlets (1000-7000) and kept PEs constant. For simulation, 100 PEs (100*2*5=1000) are considered. Values and ranges of other parameters that are used in simulation are given in Table 3. The threshold values for resource, machine and PE are 0.8, 0.75, and 0.6 respectively [7, 17, 19].

Table 3. Values of various parameters.

Type	Parameter	Range
Cloudlet	Number of cloudlets	1000-7000
	Cloudlet length	2-8 (MI)
	Deadline	1-10 (s)
Resource	Number of Resources	100
	Resource threshold	0.8
Machine	Number of machines	2
	Machine threshold	0.75
PE	Number of PE’s	5
	PE threshold	0.6

Figure 2 shows the ART of EGDC [7], EGDFR [17] and proposed model versus number of cloudlets. The ART is the time difference between task submission and first response generated by the task. As number of cloudlets increases, system load increases and ART also increases.

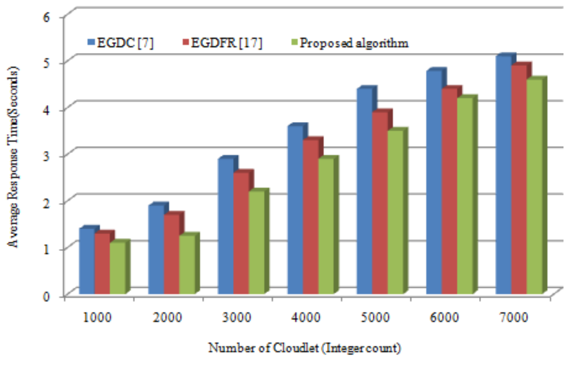

Figure 2. ART vs. no. of cloudlet.

The result shows that proposed model reduces upto 26% of the ART over other algorithms. The proposed model provides 1.25s ART when number of cloudlet is 2000 rather than 1.7s at the other algorithms. The reduction in ART is due to selection of least loaded and non-faulty resource, and calculation of checkpoint interval when assigning cloudlet to a resource for all values of number of cloudlets, makespan of proposed algorithm is always found less than EGDC and EGDFR. It is analyzed that proposed model reduces up to 29% of the makespan over other algorithms as shown in Figure 3.

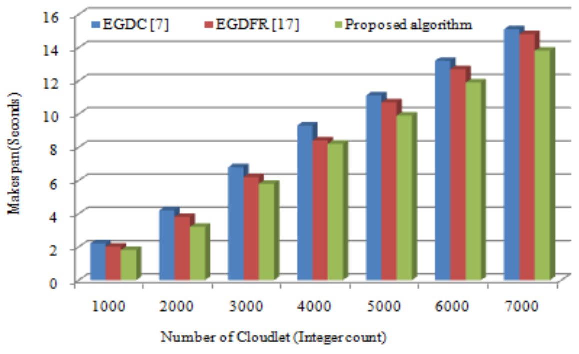

Figure 3. Makespan vs. no. of cloudlet.

The simulation result shows that proposed model increased up to 13.11% of the throughput over other algorithms when number of cloudlets is 1000 as shown in Figure 4.

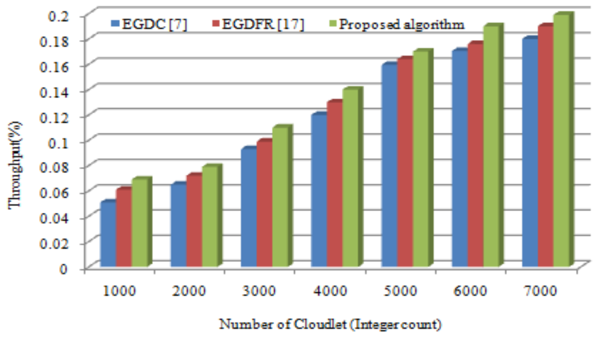

Figure 4. Throughput vs. no. of cloudlet.

Figure 5 shows the comparison of checkpoint overhead of EGDFR and proposed algorithm. The Chk-oh is calculated using Equation (6).

$$Chk\_oh = \frac{\text{makespan with checkpoints} - \text{makespan without checkpoint}}{N} \quad (6)$$

The simulation results shows that proposed model reduces up to 14% of the overhead over other algorithms. The proposed model provides 18.57% checkpoint overhead when number of cloudlets is 7000 rather than 4.28 % at the other algorithms.

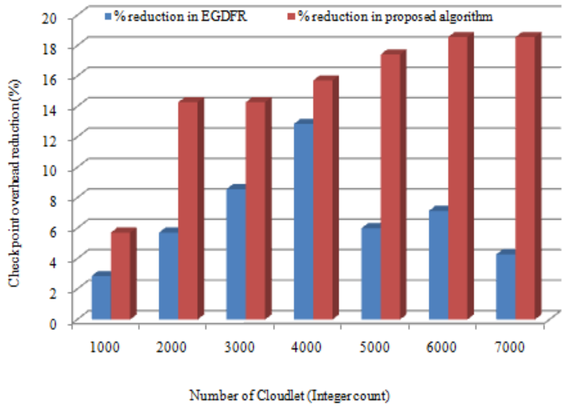

Figure 5. Checkpoint overhead vs. no. of cloudlet.

The Communication Overhead (Comm-oh) of EGDC and proposed algorithm is also determined from Equation (7). The Comm-oh is measured from Communication Time (CT) that is the time difference between latest checkpoint time on current resource and arrival time of task on another suitable resource. The proposed model provides up to 12% reduction in communication due to effective selection of resource and checkpoint interval.

$$Com\_oh = \frac{CT\ with\ checkpoints - CT\ without\ checkpoints}{N} \tag{7}$$

4.2. Case 2- Simulation By Varying Number of Pes

In this case, simulation is conducted by varying number of PEs (500-5000) and kept cloudlet constant (5000) as shown in Table 4.

Table 4. Values of various Parameters.

Type	Parameter	Range
Cloudlet	Number of cloudlets	5000
	Cloudlet length	2-8 (MI)
	Deadline	1-10 (s)
Resource	Number of Resources	500-5000
	Resource threshold	0.8
Machine	Number of machines	2
	Machine threshold	0.75
PE	Number of PE’s	5
	PE threshold	0.6
	CPE	1-5 (s)

Figure 6 shows the ART of EGDC, EGDFR and proposed model versus number of PEs. From the simulation, it is analyzed that proposed model reduces up to 21.12% of the ART over other algorithms. The proposed model provides 12.2s ART when number of PEs is 3000 rather than 15.2s at the other algorithms.

The reduction in ART is due to increased number of PEs and appropriate selection of resource.

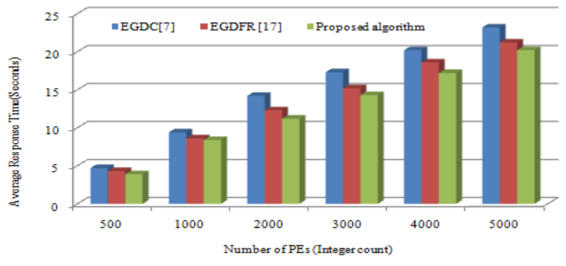

Figure 6. ART vs. no. of PEs.

The makespan of proposed algorithm is always found less than EGDC and EGDFR. The simulation result shows that the proposed model reduces up to 24% of the makespan over other algorithms as shown in Figure 7. The proposed model provides 2.01s makespan when number of PEs is 500 rather than 2.65s at the other algorithms. The simulation result shows that proposed model increased up to 25% of the throughput over other algorithms when number of PEs is 1000 as shown in Figure 8.

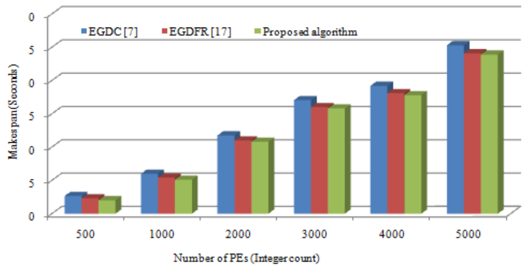

Figure 7. Makespan vs. no. of PEs.

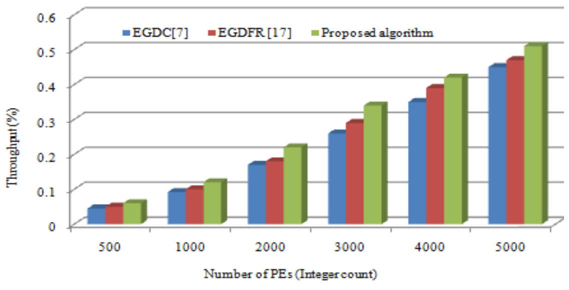

Figure 8. Throughput vs. no. of PEs.

Figure 9 shows the comparison of Chk-oh of EGDFR and proposed algorithm. The simulation results shows that proposed model reduces up to 16% of the overhead over other algorithms. The proposed model provides 28.33 % checkpoint overhead when number of cloudlets is 4000 rather than 11.66 % at the other algorithms. The proposed model provides upto 17 % reduction in communication overhead in comparison to other methods.

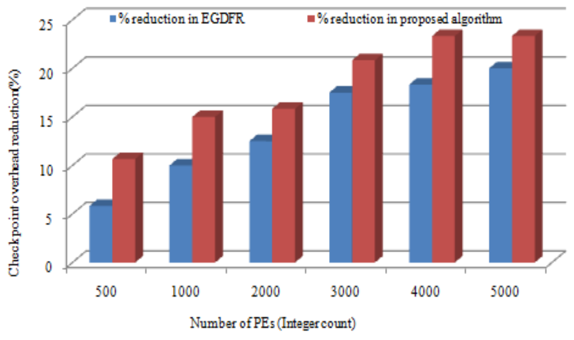

Figure 9. Checkpoint overhead vs. no. of PEs.

5. Conclusions and Future Work

Analysis of existing load balancing is carried out and classification is done based on environment, resource nature, scheduling and fault tolerance mechanisms and various optimized metrics. Based on research findings, a fault tolerance based load-balancing model has been proposes for unfinished cloudlets due to deadline constraint assigned by user. An algorithm for resource selection has been implemented also based on dynamic Resi_load and FI_Val. The comparative analysis of proposed and existing approaches has been shown in Table 5.

Table 5. Comparative analysis of Proposed and Existing approaches.

Criteria	EGDC [7]	EGDFR [17]	Proposed Model
Resource categorization criteria	Res_i_load	Res_i_load	Res_i_load and FI_Val
Cloudlets arrangement	Unsorted manner	Sorted manner	Sorted manner
ART (By varying number of cloudlet)	3.44	3.15	2.82
ART (By varying number of PEs)	14.83	13.36	12.53
Makespan(By varying number of cloudlet)	8.84	8.37	7.8
Makespan (By varying number of PEs)	13.65	12.81	12.56

The proposed model provides 22% and 13% performance improvement in terms of throughput with respect to EGDC [7] and EGDFR [17] respectively. The proposed model can be enhanced by analyzing upcoming load and distribute it more effectively.

References

[1] Arabnejad H. and Barbosa J., "A Budget Constrained Scheduling Algorithm for Workflow Applications," *Journal of Grid Computing*, vol. 12, no. 4, pp. 665-679, 2014.

[2] Calheiros R., Masoumi E., Ranjan R., and Buyya R., "Workload Prediction Using ARIMA Model and its Impact on Cloud Applications QoS," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449-458, 2014.

[3] Cao Y., Li P., and Zhang Y., "Parallel Processing Algorithm for Railway Signal Fault Diagnosis Data Based on Cloud Computing," *Future Generation Computer Systems*, vol. 88, pp. 279-283, 2018.

[4] Chang R., Lin C., and Chen J., "Selecting The Most Fitting Resource for Task Execution," *Future Generation Computer System*, vol. 27, no. 2, pp. 227-231, 2011.

[5] Garg R. and Singh A., "Fault Tolerant Task Scheduling on Computational Grid Using Checkpointing under Transient Faults," *Arabian Journal for Science and Engineering*, vol. 39, no. 12, pp. 8775-8791, 2014.

[6] Hajlaoui J., Omri M., and Benslimane D., "A Qos-Aware Approach for Discovering and Selecting Configurable Iaas Cloud Services," *Computer Systems Science and Engineering*, vol. 32, no. 4, pp. 460-467, 2017.

[7] Hao Y., Liu G., Wen N., "An Enhanced Load Balancing Mechanism Based on Deadline Control on Gridsim," *Future Generation Computer Systems*, vol. 28, pp. 657-665, 2012.

[8] Jung G. and Sim K., "Agent-Based Adaptive Resource Allocation on The Cloud Computing Environment," *in Proceeding of 40th International Conference on Parallel Processing Workshops*, Taipei, pp. 345-351, 2011.

[9] Kumar M. and Grover A., "Optimal Duty Cycling with Sleep-Wake Schedule Between Paired Nodes and Flexible Routing Across Pairs," *International Journal of Computer Applications*, vol. 144, no. 8, pp. 20-24, 2016.

[10] Lee Y., Leu S., and Chang R., "Improving Job Scheduling Algorithms in A Grid Environment," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 991-998, 2011.

[11] Liu Q., Cai W., Shen J., Fu Z., Liu X., and Linge N.,"A Speculative Approach to Spatial-Temporal Efficiency with Multi-Objective Optimization in A Heterogeneous Cloud Environment," *Security and Communication Networks*, vol. 9, no. 17, pp. 4002-4012, 2016.

[12] Mahafzah B. and Jaradat B., "The Hybrid Dynamic Parallel Scheduling Algorithm for Load Balancing on Chained-Cubic Tree Interconnection Networks," *The Journal of Supercomputing*, vol. 52, no. 3, pp. 224-252, 2010.

[13] Mahafzah B. and Jaradat B., "The Load Balancing Problem in OTIS-Hypercube Interconnection Networks," *The Journal of Supercomputing*, vol. 46, no. 3, pp. 276-297, 2010.

[14] Marimuthu P., Arumugam R., and Ali J., "Hybrid Metaheuristic Algorithm for Real Time Task Assignment Problem in Heterogeneous Multiprocessors," *The International Arab Journal of Information Technology*, vol. 15, no. 3, pp. 445-453, 2018.

[15] Masadeh R., Sharieh A., and Mahafzah B., "Humpback Whale Optimization Algorithm Based on Vocal Behavior for Task Scheduling in Cloud Computing," *International Journal of Advanced Science and Technology*, vol. 13, no. 3, pp. 121-140, 2019.

[16] Pao T. and Chen J., "The Scalability of Heterogeneous Dispatcher Based Web Server Load Balancing Architecture," *in Proceedings of International Conference on Parallel and Distributed Computing, Application and Technology*, Taipei, pp. 213-216, 2006.

[17] Patel D., Tripathy D., and Tripathy C., "An Improved Load Balancing Mechanism Based on Deadline Failure Recovery on Gridsim," *Engineering with Computers*, vol. 32, no. 2, pp. 173-188, 2016.

[18] Ramakrishna M., Kodati V., Gratz P., and Sprintson A., "GCA: Global Congestion Awareness for Load Balance in Networks-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2022-2035, 2016.

[19] Singh H. and Kumar S., "WSQ: Web Server Queueing Algorithm for Dynamic Load

Balancing," *Wireless Personal Communication*, vol. 80, no. 1, pp. 229-245, 2015.

[20] Tawfeek M., El-Sisi A., Keshk A., and Torkey F., "Cloud Task Scheduling Based on Ant Colony Optimization," *The International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 129-137, 2015.

Anju Shukla is pursuing PhD at Jaypee University of Engineering and Technology, Guna, M.P, India. She has completed B. Tech from Uttar Pradesh Technical University, Lucknow and M.Tech from Shobhit University, Meerut.

Shishir Kumar is working as Professor in the Department of Computer Science and Engineering at Jaypee University of Engineering and Technology, Guna, M.P., India. He has earned PhD in Computer Science in 20 He has 18 years of teaching and research experience.

Harikesh Singh is working as Assistant Professor in the Department of Computer Science and Engineering at Jaypee University of Engineering and Technology, Guna, M.P., India. He has earned PhD in Computer Science in 2015.