International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 12 (2018) pp. 10470-10477

Reliability Optimization Using Peer to Peer Checkpointing through Dynamic Request and Requirement Aware Mechanism

Gagan Amrit Kaur[1], and Kamaljeet Kaur[2]

[1, 2] Department of Computer Engineering & Technology, Guru Nanak Dev University, Amritsar, India.

Abstract

In cloud computing the datacenters are utilized to coordinates the distinct tasks where tasks may requires more resources and source of these cloudlets could be from thousands of users. These datacenters aims to deliver reliable services. But the equal reliability to all the users at the same time according to their requirements may be a difficult task and may vary. So in this paper we purpose a technique that considers optimized elastic reliability in cloud computing. In our scheme reliability enhancement through peer to peer check pointing with resource maximization mechanism is proposed. Resource maximization mechanism uses division policy which divides the jobs by looking at the capacity of virtual machine on which load is to be dispersed. This operation can efficiently solve the problem of reliability. It improves the resource usage in the datacenters and also gives optimized reliability to the user.

Keywords: Reliability, Checkpointing, datacenters, cloud computing

INTRODUCTION

The advanced computing is an enhancement of distributed computing models that provides services are provided on demand and has feature of pay –per –use. This type of features of cloud computing facilitate an organization to avail quick resource according to its requirement and with less service cost of cloud services. The main idea behind cloud services is to provide appealing business opportunities for developing organizations. In other words, developing organization not required to purchase the resources rather resources can be acquired on pay per use basis hence saving large amount of cost overhead on purchase of these equipments. The cloud services ought to be solid to hold the service reliability in terms of quality [1].Cloud computing contains the datacenters that hold the hosts which are further subdivided into virtual machines. Virtual machines are allotted to the clients as per requirements of the jobs. The jobs thus get the resources only if they satisfy service level agreements. This multi-tenure model [2] makes the cloud computing more inclined to different security ruptures than other appropriated computing model.

Among the most critical conditions within cloud, faults are most common. Faults once appear within the system hampers the performance of virtual machines. The job which is allotted to the virtual machine are executed on pay per use basis. The resources this allotted to the client based on service level agreement. As the faults appear SLA may be violated but cloud service provider is not penalized even if full services are not provided by the CSP. The primary reason for this distortion is faults and failures. The proactive and reactive fault tolerance provided at CSP end play a huge role in customer satisfaction and to ensure quality of service. [3].

The various abnormal situations in terms of mechanism are utilized that are categorized as detection of fault, its prevention, fault prediction and tolerance.

The fault detection is very crucial task and it initializes the fault handling technique. However because of the hesitant idea of the load oriented faults it ends up repetitive to distinguish it in the underlying state. In this manner in the past work an elastic mechanism for handling fault has been proposed which is turned out to be fit for distinguishing the faults successfully. As other techniques that deal to manage the faults at different stages either in proactive or responsive way is given. Among those the prevention of fault is imperative since it tries to keep the faults with legitimate cautious component. As often the various faults are created that would beat the current protection framework hence fault tolerance ends up essential in the cloud frameworks as a possibility for handling fault. Aside from different strategies Fault tolerance is an endeavor to guarantee benefit congruity regardless of the fault events.

There are mechanisms in cloud that uplift the reliability associated with the cloud environment. These mechanisms includes checkpointing, task resubmission, rejuvenation replication etc. in order to enhance the performance, checkpointing approach for fault tolerance is commonly used. There exists savepoints in checkpointing mechanisms. These savepoints ensure the progress monitoring and saving mechanism at back up servers so that in case of failure progress can starts from where it is left over [4]. However to keep up aggregate checkpointing for a fault inclined circumstance where failure is normal, requires exceptional checkpointing for each limited time interims. The small time interim turns out to be more space and tedious the checkpointing can be. Besides dividable jobs, for example, if there should be an occurrence of enormous data investigation if any one VM creates wrong yield it can be passed on to others and in this way the whole yield can be debased. The checkpointing in such case is repetitive and can cause overheads in performance. However there is just set number

of research work is present for improving checkpointing strategy. Checkpointing, still have issue of triggering at uniform or discrete time intervals.. Consequently checkpointing is done as a comprehensive task which implies if a failure is normal then the number of checkpoints are expanded for each VMs constantly. This means number of checkpoints at every distinct interval can be increased depending upon the level of faulty environment. Higher the faulty environment more will be the checkpoints.

Requirements and services corresponding to the organization may vary according to user requirements and the type of jobs. Faults and failures may arise due to topology break, physical deterioration of hardware components, etc. could lead to loss of data and progress made at VM. To tackle such issues fault tolerant strategies like checkpointing can be used [11][14]. Checkpointing in heavy faulty environment must be at uniform time intervals, that lead to slow progress of job processing [3][4]. In other words in case of heavy faults, checkpoints will be more significant that lead to slow execution of jobs or tasks. The problems of job allocation to virtual machines can still be optimized at VM level considering load allotment at each virtual machine [12]. The above analysis of literature suggests effectiveness of checkpoints but it may not always produce best results. However the activity is frequently divided into different energy preservation mechanisms to suit the VMs load allocation.

To tackle the issues of job allocation, scheduling mechanism is required to screen the execution of the VMs online to rate the physical server has been proposed to recognize the fitting VMs for mission basic applications. The proposed strategy with peer to peer approach along with best possible server allotment for holding backup image is proposed. This calculation adequately decides the VM faults and regularly move the errand to another working VM, at times it requires to move the whole group of undertaking that constitute the activity.

Organization of this paper is given as under: Section 2 describes the literature survey, section 3 presents the proposed system, section 4 gives the result and performance analysis, section 5 gives conclusion and future scope and last section gives the references.

LITERATURE SURVEY

Reliability of cloud is at stakes when different users interact with the cloud. In order to ensure the fault tolerance and reliability, different mechanisms are researched over. This section discusses checkpointing mechanism as a fault tolerance mechanism. Moreover, this literature [5] reveals a tool that takes over the burden of modifying hardware modules for checkpointing. Task sorting is missing in this literature.

A. Guermouche in [6] proposed a mechanism to store the checkpoint image on the server in uncoordinated manner. In other words, server selection is on the basis of storage space. Larger storage space servers can host more checkpointing images and recovery initiates in case deteriorated VM is found out. Performance however can be further improved in case server selection considers resource availability other than storage.

D. Jung, in [7] proposed a mechanism to minimize the checkpoint trials required to recover the progress made through the virtual machine. Cloudlets progress is monitored through the broker. In case deterioration is detected, progress made by VM becomes transferred to checkpoint server. Uncoordinated checkpointing approach ensures less trials and fast recovery.

The research work in [8] proposed a mechanism to minimize the energy consumption while executing the job. To minimize the energy consumption, mechanism of allocation is resource bound. This means VM are selected for allocation on the basis of resources they possess. More the resources, more chances of selection of VM exist for load allotment.

R. Rajachandrasekar in [9] proposed a power aware checkpointing mechanism. server selection in earlier work does not considered energy required to transfer the checkpoint image to the server. To tackle the issue, distance between the virtual machines is considered before migrating progress to the checkpoint server. Result in terms of energy and makespan shows improvement. Job partitioning however not considered in this approach.

M. V Santiago in [10] proposed a checkpointing strategy for minimizing the workload and enhancing fault tolerance degree. Coordination among fault tolerance servers increases the overall cost associated with the system. In order to tackle the issue similar backup images must be eliminated from the backup servers.

The research work in [11]proposed a optimal checkpointing strategy. Multiple algorithms are followed to first of all select the server for checkpointing image storage and then server selection policy for recovery server selection. The server with maximum resources is made. This decreases the overall execution time to execute the cloudlets over the cloud environment. The redundancy handling mechanism is used within the proposed system to tackle the issue of extra cost. The overall result is produced in terms of execution time and cost encountered in the execution of cloudlets.

K. N. Devi and A. Tamilarasi in [12] describes work process merchants of existing Grid Scheduling Systems are resistance instrument which causes wasteful timetables of use dispersed resources and it likewise compounds the use of different resources including system transmission capacity and computational cycles. In such models, crucial obligation, for example, resource revelation is assigned to the unified server machines, in this way they are related with surely understood impediments in regards to single purpose of disappointment, versatility and system blockage at joins that are prompting the server. With a specific end goal to conquer these issues, we

execute another approach for decentralized agreeable work process scheduling in a progressively dispersed resource sharing condition of Grids. At the point when there is a disappointment of job, it will move to another computational hub and resume from the last put away checkpoint. A Glow worm Swarm Optimization (GSO) for job scheduling is utilized to address the issue of heterogeneity in adaptation to internal failure of computational grid however Weighted GSO that beats the position refresh flaws of general GSO in a more effective way appeared amid correlation investigation. This framework underpins four sorts of adaptation to internal failure instruments, including the job movement, job retry, registration and the job replication systems additionally considering hazard nature of Grid processing condition. The hazard connection amongst jobs and hubs are characterized by the security request and then confide in level. Our assessment based recreation comes about demonstrate that our calculation has shorter makespan and more effective. We additionally investigate the productivity of the proposed approach against a brought together organized work process scheduling system and demonstrate that our approach is more effective than the incorporated strategy as for accomplishing exceedingly planned calendars.

P. Graubner in [13] proposed an approach for enhancing the vitality effectiveness of framework as-a-benefit clouds is introduced. The approach depends on performing live movements of virtual machines to spare vitality. As opposed to related work, the vitality expenses of live movements including their pre-and post-handling stages are considered, and the approach has been executed in the Eucalyptus open-source cloud registering framework by proficiently joining a multi-layered document framework and disseminated replication square gadgets. To assess the proposed approach, a few short-and long haul tests in light of virtual machine workloads created with regular working framework benchmarks, web-server copies and in addition diverse Map Reduce applications have been led. The outcomes demonstrate that vitality investment funds of up to 16 percent can be accomplished in a beneficial Eucalyptus condition.

The grid figuring in [14] empowers the clients to share the heterogeneous resources which are dispersed topographically. The fundamental preferred standpoint of grid is to use the unused resources in other words a compelling use of resources. At the point when considering about the resource usage the emphasis is on the distinctive methodologies and techniques which are actualized for successful scheduling .This paper shows a broad overview about the distinctive systems accessible in scheduling the resource.

S. Di et al. in [15] goes for advance adaptation to non-critical failure methods in light of a checkpointing/restart component, with regards to cloud processing. Our commitment is three-crease. (1) We infer a new recipe to figure the ideal number of checkpoints for cloud jobs with differed dispersions of disappointment occasions. Our investigation isn't just non specific with no presumption on disappointment likelihood circulation, yet in addition appealingly easy to apply by and by. (2) We outline a versatile calculation to upgrade the effect of checkpointing with respect to different costs like checkpointing/restart overhead. (3) We assess our advanced arrangement in a genuine bunch condition with several virtual machines and Berkeley Lab Checkpoint/Restart instrument. Errand disappointment occasions are imitated through a generation follow created on a vast scale Google server farm. Analyses affirm that our answer is genuinely appropriate for Google frameworks. Our improved recipe beats Young's equation by 3-10 percent, diminishing divider clock lengths by 50-100 seconds for every job by and large.

Checkpointing with rollback recuperation in [16] proposed a discrete voltage frequency levels for assignment of checkpointing image to the server. Checkpointing image storage and recovery considering power specified through DVFS levels enhances performance. Fast execution of cloudlets accomplishes with the proposed system. More cloudlets can be executed as the powered VM was considered for allocation of tasks. Task partitioning however is not considered that can further enhance the performance in terms of reliability.

Literature work suggests that least amount of work is done towards the partitioning of task depending upon the virtual machine availability. Performance in terms of reliability and execution time can be improved considering the job partitioning considered in the proposed system.

PROPOSED SYSTEM

Cloud is exposed to infinite users with distinct requirements. The job requires resource and if resource is not available then job must wait or job will fail. Proposed system enhances the reliability of job execution by ensuring every job although requiring more resources can be allotted to VMs with fewer resources. To accomplish this, proposed system uses the mechanism in which jobs or tasks are partitioned depending upon the resource availability. Peer to Peer checkpointing approach ensures regular backup of progress made by the virtual machine at backup machine. To take the backup field intervals are made. In other words after that fixed interval of time backup is initiated. Uniform environment indicates that the interval of back up is fixed. E.g. if initial back up time is t and α is the uniform time interval then $t+\alpha$ will be the next time interval at which back is made. Backup machine selection is on the basis of optimal resources a VM has. If the resources such as RAM, processing elements, OS, MIPS etc. associated with the virtual machines are maximum then VM is considered as optimal. Failure detection phase includes broker monitoring. Broker continues investigate the virtual machine. As the current VM deteriorates from its original metric requirements, deterioration is detected and recovery from backed up image imitated. Detailed methodology is given in this section as

Dynamic Peer to Peer Checkpointing

It will take the following steps to assign the jobs to cloud according to the requests of the clients:

Step 1: Jobs are received from the user and stored within the joblist.

$$Job_{List_i} = User\_jobs_i$$

User_job variable defines a list in which jobs corresponding to users are stored.

Step 2: Splitting of jobs into tasks according to resource availability. Resources considered are RAM, PEs and Image size.

$$Task_{list_i} = \frac{Jobs_i}{N}$$

Jobs indicate the resource requirement of the jobs and N is the availability. E.g. if N=10 and jobs=20 then current job is partitioned into 10 distinct tasks that can be executed on current virtual machine parallel.

Step 3: After the allotment of jobs to VMs, the peer checkpointing commences. VM is directly associated with the broker hence broker take the jobs checks the VM resources and allocate the job to the Virtual machines.

Identification of optimal server having the maximum resources for checkpointing image storage

For i=1: n

For j=i: n-i-1

If($server_{j.resources}$>$server_{j+1.resources}$)

Temp= $server_{j.resources}$

$server_{j.resources=}$ $server_{j+1.resources}$

$server_{j+1.resources}$=temp

end of if

end of for

end of for

Temp variable ultimately gives the optimal server for image storage.

Step 4: Time interval is uniform after which checkpointing is initiated automatically. Here Threshold_Time is SLA_bound

If(T_i>Threshold_Time)

Initiate checkpoint mechanism for image storage

Server[Temp]=Progressive_VM_Image

end of if

Threshold_Time here describes the minimum value associated with the checkpointing time interval. E.g. if threshold is 5 seconds and time start at 0, then at every 5 seconds checkpointing continuously takes place.

Temp variable gives the address of optimal checkpoint server where progressive VM (VM performing task) content is placed.

Step 5: Monitoring phase

This phase is used to check the VM for deterioration. This phase is broker oriented where broker keeps a eye on virtual machine operation. Deadline is the interval in which job must be executed. In other words, these jobs are time critical jobs.

If(VM[i].Execution_Time>Job[i].Deadline && VM[i].Temp>Threshold_temp)

Deteriorated VM is detected then recovery VM is selected and checkpoint image is copied from optimal server to VM[i].

Set i=i+1

VM[i]=Server[Temp]

End of if

Deteriorated Virtual machine once detected, next VM from VM list is selected to place image from the recovery server. Here Temp variable gives the location of optimal server where image of progressive machine is placed.

Step 6: If there is no failure then check for time slot which if arrives, progress is stored within the optimal VM. To find the optimal VM for checkpoint image storage step 3 is performed.

The flow of proposed system is given the figure 1. The result of the proposed work is improvement in terms of reliability. Reliability is specified in terms of number of cloudlets submitted to the number of cloudlets successfully executed by the cloud environment. Metric used in the proposed system are execution time and reliability. Execution time is given as equation 1.

$$Execution_{time} = Finish_{time} - start_{time} \quad (1)$$

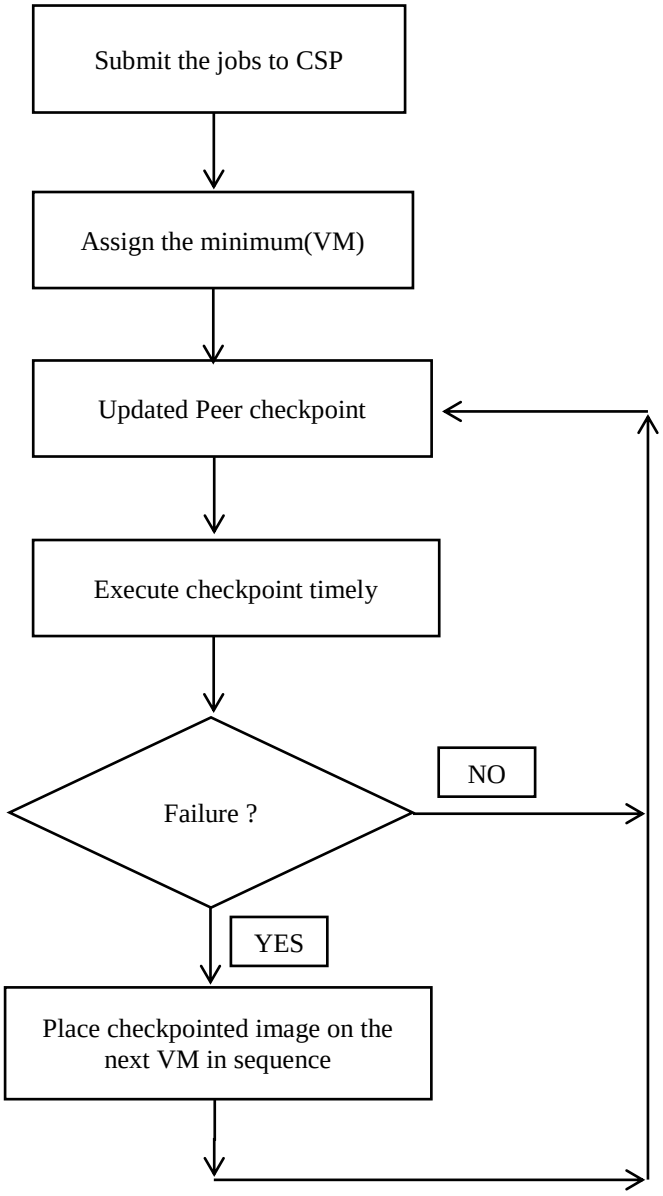

Figure 1: Peer to Peer Checkpointing with Task division approach for reliability enhancement

Reliability is calculated in terms of number of jobs successfully executed to the number of jobs submitted. It is evaluated using the following equation (2).

$$Reliability = \frac{Jobs_{Successfully_{Executed}}}{Total_{jobs}} \quad (2)$$

The affect of proposed system is elaborated in the result section. Result and performance analysis consist of two scenarios. Comparative analysis of two different scenarios is made. In first results are evaluated when the system is fault free. In second situation, faulty environment is considered for checking the performance of existing and proposed system. Byzantine fault is introduced within single progressive VM and performance is monitored. Fault injection continues to check resilience nature of the proposed system.

Scenario 1: First scenario presents result when no faults are injected within the system

Scenario 2: Second scenario presents the result in case of faulty environment.

RESULT AND PERFORMANCE ANALYSIS

This section described the results obtained through the proposed approach. The results obtained are compared against the existing approach to prove the validity of the proposed work. Simulation is conducted within CloudSim 4.0 with the integration of NetBeans 8.0. Parameters used in the proposed work includes

Execution time: Total time required to execute entire cloudlets. It is defined by the following equation.

$$\boldsymbol{Execution_{Time} = Finsih_{Time} - Start\_time}$$

Execution time of existing and proposed system shows deviation, however proposed system shows little deviation from fault free environment. The execution time obtained before and after the fault injection is given as under:

Table 3: Execution time of existing and proposed system with and without faulty environment

Number of tasks	Without Faulty Environment		With Faulty Environment	
	Peer to Peer Checkpointing	Dynamic Peer to Peer Checkpointing	Peer to Peer Checkpointing	Dynamic Peer to Peer Checkpointing
100	1768.94	1523.33	1968.94	1723.33
200	3136.6	3033.2	3836.6	3533.2
300	5499.57	5244.43	5999.57	5744.43
400	5967.07	5524.55	6067.07	5824.55
500	8278.25	7944.23	9278.25	8244.23

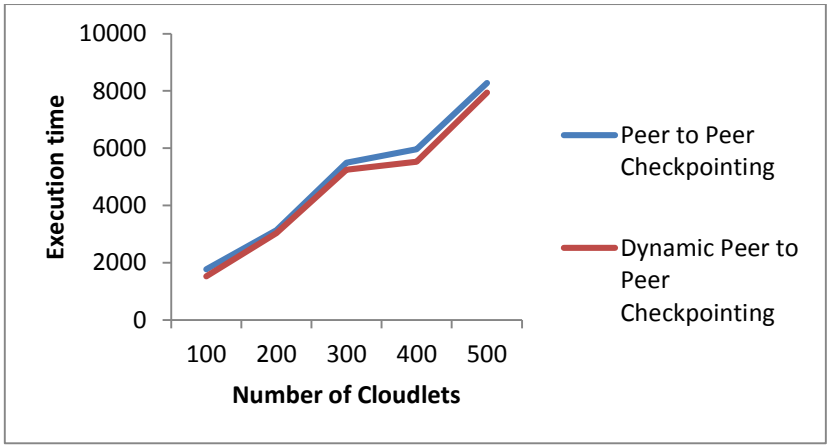

Figure 2: Graph depicting execution time without introduction of faults

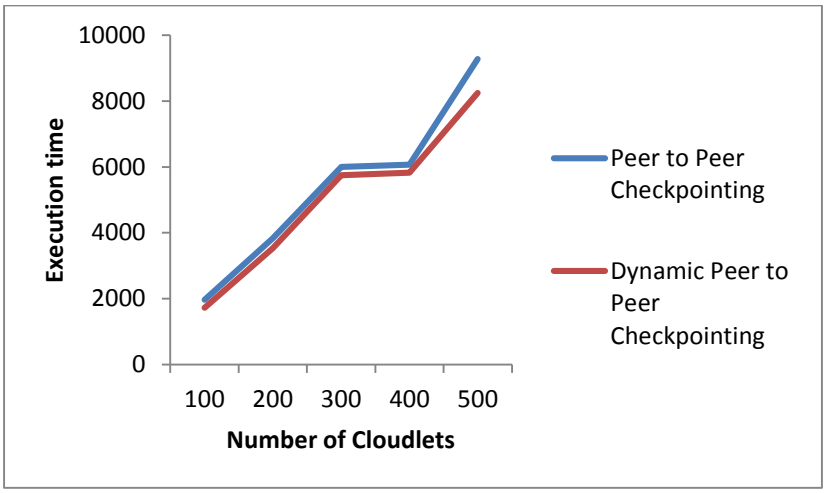

Figure 3: Graph depicting execution time in faulty environment

Reliability: It is calculated by observing number of cloudlets successfully executed by the cloud machines.

$$\text{Reliability} = \frac{Jobs_{Executed}}{Number_{of_{submitted}}}$$

Reliability indicates number of cloudlets successfully executed by the virtual machine. Virtual machines are continuously monitored by the broker. In case deterioration is detected, recovery server place the image to next virtual machine from the VM list and cloudlet execution continues.

Reliability also suffers a setback when faults are injected within the system. In other words cloudlets execution deteriorated when faults are injected within the system. Results in terms of reliability of proposed system show little deviation showing worth of the study. Reliability in case of fault free and faulty environment is given as under:

Table 2: Reliability in case of existing and proposed system with and without fault introduction

Number of tasks	Without Faulty Environment		With Faulty Environment	
	Peer to Peer Checkpointing	Dynamic Peer to Peer Checkpointing	Peer to Peer Checkpointing	Dynamic Peer to Peer Checkpointing
100	0.7	0.9	0.5	0.75
200	0.75	0.9	0.55	0.77
300	0.8	0.87	0.58	0.77
400	0.7	0.9	0.62	0.76
500	0.8	0.9	0.65	0.77

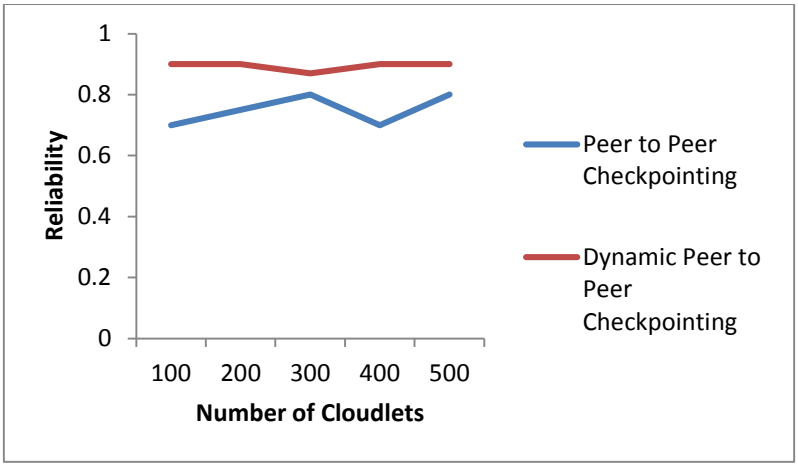

Figure 4: Graph depicting reliability in fault free environment

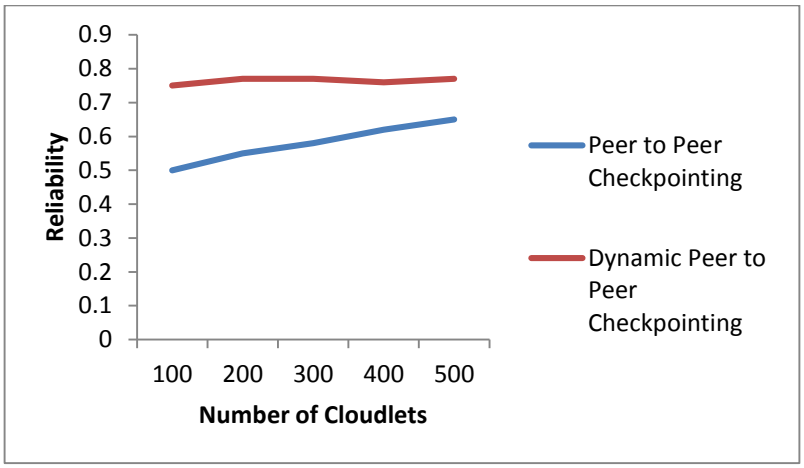

Figure 5: Graph depicting reliability in faulty environment

As the faults are introduced within the system, reliability decreases. But the proposed system shows better finish rate as compared to existing system.

CONCLUSION

This paper proposes a novel way to deal with giving versatile dependability advancement in cloud computing. Depending on external and internal checkpointing, the issue of joint reliability is defined as advancement, in which data center need to discover checkpoint scheduling and make directing/position choices with a specific end goal to amplify a total utility of dependability. Job partitioning approach merged with peer to peer checkpointing approach yield better results and finishes more cloudlets in short intervals as compared to existing system. The experimental result shows that the reliability is improved. The reliability can be collaborated with energy efficiency and real time cloud environment can be considered in future work.

REFERENCES

[1] A. E. El-Desoky, H. A. Ali, and A. A. Azab, "Improving fault tolerance in desktop grids based on incremental checkpointing," *2006 Int. Conf. Comput. Eng. Syst. ICCES'06*, pp. 386–392, 2006.

[2] G. Aupy, A. Benoit, M. E. M. Diouri, O. Glück, and L. Lefèvre, "Energy-Aware Checkpointing Strategies," pp. 279–317, 2015.

[3] J. S. Plank, K. Li, and M. A. Puening, "Diskless checkpointing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 10, pp. 972–986, 1998.

[4] S. Chinnathambi, A. Santhanam, J. Rajarathinam, and M. Senthilkumar, "Scheduling and checkpointing optimization algorithm for Byzantine fault tolerance in cloud clusters," *Cluster Comput.*, pp. 1–14, 2018.

[5] D. Koch, C. Haubelt, and D.- Erlangen, "Efficient Hardware Checkpointing Concepts , Overhead Analysis , and Implementation," pp. 188–196, 2007.

[6] A. Guermouche, T. Ropars, E. Brunet, M. Snir, and F. Cappello, "Uncoordinated checkpointing without

domino effect for send-deterministic MPI applications," *Proc. - 25th IEEE Int. Parallel Distrib. Process. Symp. IPDPS 2011*, pp. 989–1000, 2011.

[7] D. Jung, S. Chin, K. Chung, and H. Yu, "An Efficient Checkpointing Scheme Using Price History of Spot Instances in Cloud Computing Environment," *IFIP*, pp. 185–200, 2011.

[8] G. Aupy, A. Benoit, R. Melhem, P. Renaud-Goud, and Y. Robert, "Energy-aware checkpointing of divisible tasks with soft or hard deadlines," *2013 Int. Green Comput. Conf. Proceedings, IGCC 2013*, 2013.

[9] R. Rajachandrasekar, A. Venkatesh, K. Hamidouche, and D. K. D. K. Panda, "Power-Check : An Energy-Efficient Checkpointing Framework for HPC Clusters," *IEEE Access*, 2015.

[10] M. V Santiago, S. E. P. Hernández, L. A. M. Rosales, and H. H. Kacem, "Checkpointing Towards Dependable Business Processes," vol. 14, no. 3, pp. 1408–1415, 2016.

[11] A. Zhou, Q. Sun, and J. Li, "Enhancing Reliability via Checkpointing in Cloud Computing Systems," *IEEE*, pp. 108–117, 2017.

[12] K. N. Devi and A. Tamilarasi, "IMPROVING FAULT TOLERANT RESOURCE OPTIMIZED AWARE JOB SCHEDULING FOR GRID COMPUTING," vol. 10, no. 5, pp. 763–773, 2014.

[13] P. Graubner, M. Schmidt, and B. Freisleben, "Energy-efficient Management of Virtual Machines in Eucalyptus," 2011.

[14] V. Sumathy, "A Detailed Study of Resource Scheduling and Fault Tolerance in Grid," vol. 8, no. 6, pp. 357–361, 2011.

[15] S. Di *et al.*, "Optimization of Cloud Task Processing with Checkpoint-Restart Mechanism," 2013.

[16] M. Salehi, M. K. Tavana, S. Rehman, S. Member, M. Shafique, and A. Ejlali, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," pp. 1–12, 2016.